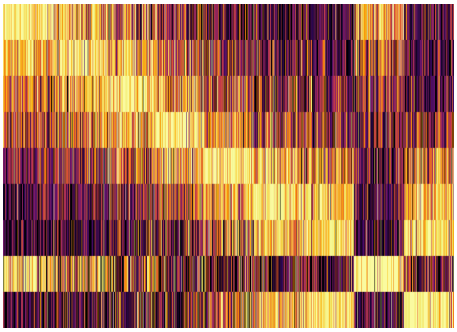
Manifold-Based Sensorimotor Representations for Bootstrapping of Mobile Agents



Carsten Rachuy

December 2019

Dissertation zur Erlangung des akademischen Grades eines
Doktors der Ingenieurwissenschaften

Dr.-Ing.

Universität Bremen - Fachbereich 3 Mathematik/Informatik

Title figure: Composite result matrix, displaying the evaluation results from one hundred invocations of the bootstrapping classification algorithm for a hypothesis set consisting of nine hypotheses. Each entry visualizes the intrinsic conflict, that is associated to a sensorimotor map, created by a hypothesis-specific folding of the sensorimotor chain. Each chain was generated by an agent, that executed a random sequence of fifty controls, utilizing a nonholonomic actuator for traversing a two-dimensional environment. This matrix allows one, to visually infer the performance of the bootstrapping classification algorithm: The better, i.e., the more robust the results of the classifier are, the brighter the diagonal of this matrix and the darker the rest of the matrix entries will be. Figure taken as an example from the evaluation conducted in Scenario V, Section 7.4.5.7.

Date of Submission

17 December 2019

Date of Defense

17 March 2020

Reviewers

Prof. Dr. Kerstin Schill, University of Bremen, Germany

Prof. Dr.-Ing. Udo Frese, University of Bremen, Germany

Abstract

Systems that are able to move autonomously through their environment require consistent map representations that serve as a foundation for navigational tasks like localization, motion planning, and motion control. A fundamental aim of these kinds of representations is thus to represent the geometry of the environment and relate it to sensory inputs and motor controls. As navigation is a task technical as well as biological systems encounter frequently, systems in both domains have to meet the challenge of establishing suitable map representations that capture the relationship between sensory and motor properties.

Within the domain of mobile robotics, various approaches to endow systems with the ability of navigation exist. However, most of them require prior knowledge to be provided to the system in terms of sensor and motion models that often explicitly model the geometric relation between sensor inputs, the outcome of controls, and the map representation itself. While approaches to learning sensor parameters and motion models exist, these are often build upon a priori knowledge and are thus tailored to calibrate and fine-tune known models to specific sensors or actuators. Moreover, the synthesis of controls is usually not a native trait of the map representation, but requires domain specific operations and transformations.

With respect to complex biological entities, the ability to perceive their environment, localize themselves within their natural habitat, and deliberately navigate is a common trait. Even in simple entities, a neural coupling between sensory and motor components can be observed. In this regard, perception processes and means of interaction with the environment show a great deal of diversity. However, the nervous system exhibits structural similarities across many types of animals, especially with respect to the neuron, the atomic unit of biologic information processing. The question arises whether the neurobiological representation of sensory inputs, motor controls, and the fundamental structure of the environment are genetically encoded in complex biological entities or created by exploration of their surroundings and utilization of their abilities. Another question is how information on geometric relations is encoded, to what extent it is already present, and to what extent it is learned from suitably interpreting perceptions and actions.

The main subject of this thesis is the development of a domain-independent algorithm that allows an autonomous system to process sequences of the sensorimotor interaction with its environment and to create a consistent representation, ultimately allowing it to assign a geometric interpretation to its motor capabilities. On an abstract level, both technical and biological autonomous systems interact with their environments by either using their sensors to perceive information on their current state or by using their actuators to change their state, i.e., move through the environment. This process creates an alternating sequence of sensory inputs and motor controls which we utilize as a foundation for developing a joint sensorimotor represen-

tation that is characterized by a tight coupling of sensory and motor properties. To this end, we utilize manifolds, mathematical structures that locally resemble n-dimensional Euclidean space whose global structure is, however, not bound to this restriction. More specifically, we use Lie groups, smooth manifolds endowed with a group structure, that allow for an elegant representation of geometric operations as a central foundation for a sensorimotor representation. Expressing motor controls with respect to the manifold structure allows us to transform the sensorimotor interaction sequence into a specific set of data points of perceived samples. The geometric relations between these samples are dependent on the manifold, i.e., the transformations it induces. Applying a suitable conflict function to these data points allows us to assign a measure of intrinsic conflict to the entire set and thus, ultimately, to the transformation. Finding a manifold and a transformation that minimizes this intrinsic conflict function corresponds to finding a topological structure that is the best fit for expressing the sensorimotor space the entity resides in.

It is shown that a consistent, sensorimotor representation can be constructed, utilizing only sensory inputs and motor controls accessible to the entity where no specific interpretation to either of these modalities is provided to the system in advance. Experiments in a virtual environment are conducted that show the applicability of the approach with respect to different sensor and motor configurations.

Zusammenfassung

Systeme, denen es möglich ist, sich autonom durch ihre Umgebung zu bewegen, benötigen konsistente Kartenrepräsentationen, die als Grundlage für Navigationsaufgaben wie Lokalisierung und Bewegungsplanung und -steuerung dienen können. Ein grundlegendes Ziel dieser Art von Repräsentationen ist somit, die Geometrie der Umgebung zu repräsentieren und sie sowohl in Bezug zu Sensormessungen als auch zu Motorkommandos zu setzen. Da sowohl technische als auch biologische Systems oft mit Navigationsaufgaben konfrontiert sind, müssen sich in beiden Domänen Systeme der Herausforderung stellen, angemessene Kartenrepräsentationen aufzubauen, die den Zusammenhang zwischen Sensordaten und Motorkommandos repräsentieren.

Im Bereich der Robotik existieren verschiedene Ansätze, um Systeme mit der Fähigkeit zur Navigation auszustatten. Die meisten dieser Ansätze erfordern jedoch, dass dem System Vorwissen in Form von Sensor- und Bewegungsmodellen zur Verfügung gestellt wird, welche oftmals explizit die geometrischen Zusammenhänge zwischen Sensordaten, Motorkommandos und der Kartenrepräsentation modellieren. Obwohl Ansätze existieren, Sensorparameter und Bewegungsmodelle zu lernen, beziehen sich diese oft auf vorhandenes Wissen und sind daher eher darauf ausgelegt, bereits bekannte Modelle zu kalibrieren und so auf spezifische Sensorik oder Aktuatorik abzustimmen. Des Weiteren erfolgt die Synthese von Motorkommandos meist nicht nativ aus der Repräsentation heraus, sondern erfordert domänenspezifische Operationen und Transformationen.

Ein gemeinsames Merkmal komplexer biologischer Systeme ist deren Fähigkeit, ihre Umwelt wahrzunehmen, sich innerhalb ihres natürlichen Lebensraumes zu lokalisieren und gezielt zu navigieren. Bereits in einfachen Systemen ist hierbei eine neuronale Kopplung sensorischer und motorischer Komponenten zu beobachten. Weisen die in der Natur beobachtbaren Wahrnehmungsapparate und Interaktionsmöglichkeiten eine große Diversität auf, so ist das Nervensystem, insbesondere hinsichtlich der elementaren Einheit der biologischen Informationsverarbeitung, dem Neuron, in vielen Tieren ähnlich aufgebaut. Es stellt sich somit die Frage, inwiefern die neurobiologische Repräsentation von Sensordaten, Motorkommandos und der grundlegenden Struktur der Umgebung genetisch kodiert ist oder durch Exploration und Nutzung der eigenen Fähigkeiten erstellt wird. Eine weitere Frage ist, wie Informationen über geometrische Zusammenhänge kodiert werden, inwiefern diese bereits vorliegen und zu welchem Grad sie durch geeignete Interpretation von Wahrnehmungen und Aktionen gelernt werden.

Zentrales Thema dieser Arbeit ist die Entwicklung eines domänenunabhängigen Algorithmus, der es einem autonomen System ermöglicht, die durch die Interaktion mit seiner Umgebung generierten sensomotorischen Sequenzen zu verarbeiten und eine konsistente Repräsentation zu erstellen, die das System letztendlich in die Lage versetzt, eine geometrische Interpretation seiner motorischen Aktionen zu finden. Auf einer abstrakten Ebene interagieren sowohl

technische als auch biologische autonome Systeme mit ihrer Umgebung, indem sie entweder ihre Sensoren nutzen, um Informationen über ihren aktuellen Zustand wahrzunehmen, oder ihre Aktuatoren nutzen, um diesen Zustand zu verändern, d.h. um sich durch ihre Umgebung zu bewegen. Dieser Prozess generiert eine alternierende Sequenz von Sensordaten und Motorkommandos, welche als Grundlage zur Entwicklung einer sensomotorischen Repräsentation genutzt wird, die insbesondere durch die enge Kopplung von Sensor- und Motoreigenschaften charakterisiert wird.

Zu diesem Zweck werden Mannigfaltigkeiten genutzt, mathematische Strukturen, die lokal einem n-dimensionalen euklidischen Raum entsprechen, global jedoch nicht dieser strukturellen Einschränkung unterliegen. Konkret werden Lie-Gruppen als zentrale Grundlage der Repräsentation verwendet. Dies sind differenzierbare Mannigfaltigkeiten, die zusätzliche Gruppeneigenschaften aufweisen und die mathematisch elegante Repräsentation geometrischer Operationen ermöglichen. Die Nutzung von Mannigfaltigkeiten zur Repräsentation von Motorkommandos ermöglicht es, die sensomotorische Interaktionssequenz in eine Menge von Datenpunkten aus sensorischen Eingaben umzuwandeln, wobei die geometrischen Relationen zwischen einzelnen Elementen abhängig von der Mannigfaltigkeit bzw. der durch diese induzierten Transformationen sind. Durch die Nutzung einer geeigneten Konfliktfunktion kann der intrinsische Konflikt dieser Datenpunkte und somit auch der Transformation bestimmt werden. Eine Mannigfaltigkeit und eine Transformation zu finden, die diesen intrinsischen Konflikt minimieren, bedeutet, die topologische Struktur zu finden, die die beste bereinstimmung mit der sensomotorischen Umgebung, in der die Entität sich aufhält, aufweist.

Es wird gezeigt, dass eine konsistente, sensomotorische Repräsentation lediglich auf Basis der dem System zu Verfügung stehenden Sensordaten und Motorkommandos erstellt werden kann, ohne diesen eine spezifische Interpretation dieser Modalitäten zur Verfügung zu stellen. Mittels Experimenten in einer virtuellen Umgebung wird die Anwendbarkeit des Ansatzes auf verschiedene Sensor- und Aktuatorkonfigurationen gezeigt.

Acknowledgements

First and foremost, I want to thank my supervisor Prof. Dr. Kerstin Schill for her support, for introducing me to the topic of sensorimotor systems, and for giving me the freedom and the opportunities to embark on this journey. Furthermore, I would like to thank Prof. Dr.-Ing. Udo Frese for his guidance, insightful feedback, and many helpful suggestions.

I want to thank my colleagues of the Cognitive Neuroinformatics group for the warm and friendly work atmosphere, and for many fruitful discussions. I especially want to express my gratitude towards Joachim Clemens, Anne Schattel, Dorothea Brckner and David Nakath for their helpful advise, valuable input and many constructive comments.

I want to express my warmest thanks to my parents Karin Rachuy and Dieter Rachuy and my sister Johanna Rachuy for their unconditional love, their encouragement and their never-ending support during all these years. Finally, I wholeheartedly thank Verena Schmidt for her love, warmth, and her constant support and patience.

Contents

Abstract iii

Zusammenfassung v

Acknowledgements vii

Mathematical Notation xxi

1 Introduction 1
- 1.1 Rational Agents . . . 2
- 1.2 Robot Navigation . . . 5
- 1.3 Motivational Example . . . 6
- 1.4 Motivation . . . 9
  - 1.4.1 Representations for Technical Systems . . . 10
  - 1.4.2 Representations in Biological Systems . . . 11
  - 1.4.3 Biology as Inspiration . . . 12
- 1.5 Aim and Approach . . . 12
  - 1.5.1 Represention . . . 13
  - 1.5.2 Bootstrapping . . . 14
- 1.6 Research Questions and Contributions . . . 15
- 1.7 Structure of the Thesis . . . 16

2 State of the Art 17
- 2.1 Biology . . . 17
  - 2.1.1 Neurobiological and Evolutionary Preliminaries . . . 18
  - 2.1.2 Entorhinal Cortex and Hippocampus . . . 18
  - 2.1.3 Sensory and Sensorimotor Integration . . . 19
  - 2.1.4 Adaption . . . 19
  - 2.1.5 Cognition and Psychology . . . 20
- 2.2 From Biology to Technical Systems . . . 20
  - 2.2.1 Neural Networks . . . 21
  - 2.2.2 Particle Swarm Optimization . . . 21
  - 2.2.3 Motor Babbling . . . 21
  - 2.2.4 Active Exploration . . . 22
- 2.3 Fundamental Concepts . . . 23

2.3.1 Map Representation and Navigation . . . 23
2.3.2 Bootstrapping . . . 25
2.3.3 Manifolds . . . 27
2.3.4 Lie Groups . . . 27
2.3.5 Molecular Docking and Protein Folding . . . 28
2.4 Related Concepts . . . 28
2.4.1 Navigation . . . 29
2.4.2 Autonomous Systems and Robotics . . . 30

3 Mathematical Foundations **33**
3.1 Synopsis . . . 33
3.2 Configuration Space . . . 35
3.3 Manifolds . . . 37
3.3.1 Paths on Manifolds . . . 40
3.3.2 Parameterization of Paths . . . 42
3.4 Tangent Spaces . . . 43
3.5 Lie Groups . . . 46
3.6 Lie Algebras . . . 48
3.7 Holonomic and Nonholonomic Systems . . . 50
3.8 Summary . . . 53

4 Sensorimotor Agents **55**
4.1 Synopsis . . . 55
4.2 Preliminary Definitions and Assumptions . . . 56
4.3 Extrinsic View . . . 59
4.3.1 Configuration Space . . . 60
4.3.2 Control Space . . . 61
4.3.3 Transition Function . . . 62
4.3.4 Sample Set . . . 70
4.3.5 Percept and Sample Function . . . 71
4.4 Intrinsic View . . . 73
4.4.1 Sensor Invocation . . . 73
4.4.2 Actuator Invocation . . . 73
4.4.3 Invoking the Sensorimotor Map . . . 75
4.4.4 Intrinsic Agent Loop . . . 75
4.5 Summary . . . 76

5 Sensorimotor Maps **79**
5.1 Synopsis . . . 80
5.2 Data Structures and Topological Relations . . . 81
5.2.1 The Sensorimotor Chain . . . 81
5.2.2 Sensorimotor Neighborhoods . . . 85
5.2.3 Reference Frame Transformations . . . 95
5.3 Building Sensorimotor Maps . . . 104

5.3.1 Motivational Example . . . 104
5.3.2 The Sensorimotor Graph . . . 108
5.3.3 The Sensorimotor Map . . . 111
5.4 Summary . . . 121

6 Bootstrapping using Sensorimotor Maps 123
6.1 Synopsis . . . 123
6.2 Preliminary Definitions . . . 125
6.2.1 Parameterization of the Sensorimotor Map . . . 125
6.2.2 Intrinsic Consistency and Conflict in Sensorimotor Neighborhoods . . . 128
6.3 Bootstrapping Algorithms . . . 138
6.3.1 Bootstrapping as a Classification Problem . . . 138
6.3.2 Bootstrapping as an Optimization Problem . . . 140
6.4 Revisiting the Agent Loop . . . 145
6.5 Bootstrapping using Sensorimotor Maps . . . 147
6.5.1 Classification . . . 147
6.5.2 Optimization . . . 149
6.6 Summary . . . 152

7 Evaluation 153
7.1 Synopsis . . . 153
7.2 Evaluation Parameters . . . 155
7.2.1 Parameter List . . . 155
7.2.2 Simulation Environment . . . 156
7.2.3 Agent Configuration . . . 157
7.2.4 Exploration Patterns . . . 161
7.2.5 Sensorimotor Map and Bootstrapping . . . 163
7.2.6 Hypotheses . . . 166
7.3 Preliminary Definitions . . . 169
7.3.1 Simulation and Agent Initialization . . . 169
7.3.2 Visualization of Results and Notation . . . 170
7.4 Experiments . . . 175
7.4.1 Scenario I : Conflict Function Evaluation . . . 175
7.4.2 Scenario II : Holonomic and Nonholonomic Pattern Evaluation on SE(2) 179
7.4.3 Scenario III : Holonomic and Nonholonomic Pattern Evaluation on SE(3) 183
7.4.4 Scenario IV : Holonomic and Nonholonomic Baseline . . . 185
7.4.5 Scenario V : Holonomic and Nonholonomic Classification . . . 191
7.4.6 Scenario VI : Holonomic and Nonholonomic Optimization . . . 208
7.5 Summary of Experiments . . . 219

8 Conclusion 221
8.1 Summary and Discussion . . . 221
8.2 Outlook . . . 224

A Prominent Lie Groups **227**

B Mobile Robot Example **231**

C List of Hypotheses **235**

D Additional Tables and Figures **241**

Bibliography **315**

List of Figures

1.1 Conceptual layout of a rational agent. 4
1.2 Layout of a simple robotic system and the environment it is situated in. 6
1.3 Paths the robot traveled in Task I, Task II and Task III. 9

3.1 Examples for robotic systems and their configuration spaces. 35
3.2 Relation between the entity-specific occupancy function and the configuration space. 37
3.3 Visualization of S^2 as an example for a manifold. 38
3.4 Coordinate transformation between two overlapping neighborhoods U_α and U_β. 39
3.5 Depiction of paths on the manifold Q^n. 41
3.6 Partially obstructed path on the configuration space. 42
3.7 Visualization of tangent vectors and tangent spaces on the smooth manifold Q^n. 43
3.8 Vector field $X : Q^n \to TQ^n$ on the smooth manifold Q^n. 44
3.9 Comparison of a holonomic vaccuum cleaning robot and a car-like nonholonomic system. 52
3.10 Visualization of the Lie bracket operation for commuting and non-commuting controls that correspond to the generators of the special Euclidean group $SE(2)$. 53

4.1 Extrinsic view of the sensorimotor system. 59
4.2 Relationship between the control space C, the set of valid controls $C_\odot$, and the extended control space C_*. 63
4.3 Transition function, modeled as a composition of τ_*, $\tau_\cup$, the exponential map and the group operation. 66
4.4 Partial transition function with feedback. 68
4.5 Depiction of the sample function Π and the relation between the configuration space Q^n and the sample set S. 72
4.6 Intrinsic view of the sensorimotor system. 74

5.1 Visualization of the sensorimotor sequence L_t. 82
5.2 Relation between the intermediate memory node ω_i and the configuration space Q^n. 84
5.3 Visualization of the motor cover $\mathcal{R}_\odot$ and the motor neighborhood $\mathcal{M}_\odot[q_i]$. . . . 87
5.4 Visualization of the sensorimotor sample neighborhood $\Lambda_{(\mathcal{S},\odot)}[q_i]$ and the sensorimotor obstruction neighborhood $\Lambda_{(\mathcal{B},\odot)}[q_i]$. 89

5.5 Example for sensorimotor sample and obstruction sets associated to the memory node ω_i. 90
5.6 Visualization of the sensorimotor neighborhood of a nonholonomic robot using a slip/skid steering configuration. 91
5.7 Visualization of the extended motor neighborhood $\mathcal{M}_{\circledcirc}[q_i]$. 92
5.8 Visualization of the extended sensorimotor sample neighborhood $\Lambda_{(\mathcal{S},\circledcirc)}[q_i]$ and the extended sensorimotor obstruction neighborhood $\Lambda_{(\mathcal{B},\circledcirc)}[q_i]$. 94
5.9 Example for extended sensorimotor sample and obstruction sets associated to the memory node ω_i. 95
5.10 Visualization of the reference frame transformation between the two directly connected extended sensorimotor neighborhoods $\Lambda_{\circledcirc}[q_i]$ and $\Lambda_{\circledcirc}[q_j]$. 98
5.11 Difference between the reference frame transformation $\Lambda_{\circledcirc}[q_i \triangleleft q_j]$ and the projection set $\Lambda_{\circledcirc}[q_i \uplus q_j]$. 99
5.12 Calculation of the transformation between two arbitrary memory nodes ω_i and ω_j, where we assume that $i < j$. 101
5.13 Visualization of the sensorimotor chain Ω_{27} after exploring the environment and reaching the configuration with the highest temperature. 105
5.14 Direct navigation on the sensorimotor chain Ω_{27}. 106
5.15 Visualization of the effect the radius $\epsilon_{\circledcirc}$ has on the structure of the graph created by calculating the projection sets $\Lambda_{\circledcirc}[\omega_i \uplus \Omega_{27}]$ for all memory nodes in the sensorimotor chain Ω_{27}. 107
5.16 Visualization of paths on two different sensorimotor graphs, created using different values for the neighborhood radius $\epsilon_{\circledcirc}$. 108
5.17 Successive creation of the anchor graph Ξ over eight time steps. 113
5.18 The sensorimotor map, interpreted as a parameterizable atlas-like structure. . . 122

6.1 Visualization of the folded sensorimotor chain Ω_t, for four different parameterizations. 129
6.2 Illustration of the intrinsic conflict function for two different hypotheses. 132
6.3 Visualization of the difference between an adaptive and a non-adaptive strategy for executing exploration patterns in an environment with obstacles. 148

7.1 Conceptual view of the simulation environment and two example configurations. 156
7.2 Visualization of sensor feedback for the proximity and temperature sensors for the environment configuration given in Fig. 7.1c. 159
7.3 Visualization of sensor feedback for the light and range sensors for the environment configuration given in Fig. 7.1c and for two different orientations of the agent. 160
7.4 Examples for different exploration patterns for an actuator with two degrees of freedom. 162
7.5 Visualization of the effect the selection of `conflict_subsets` has on the computational cost for the conflict function Δ_{NW}. 165
7.6 Confusion matrices for a scenario initialized with hypotheses from $\mathrm{H}_1^{\oplus}\mathrm{SE}(2)$ and 5 iterations. 171

7.7 Partial result matrices for a scenario with five iterations and an agent initialized with hypothesis $H_9 \in \mathrm{H}_1^{\oplus}\mathrm{SE}(2)$. 172
7.8 Normalized partial result matrices for a scenario with five iterations and an agent initialized with hypothesis $H_9 \in \mathrm{H}_1^{\oplus}\mathrm{SE}(2)$. 173
7.9 Composite result matrices for a scenario with five iterations which is initialized with hypotheses from $\mathrm{H}_1^{\oplus}\mathrm{SE}(2)$. 174
7.10 Setup for Scenario I, conflict function evaluation. 176
7.11 Results from Scenario I, conflict function evaluation. 178
7.12 Example composite matrix from Scenario II, pattern and step size evaluation. . 182
7.13 Results from Scenario V, subset conflict calculation. 194
7.14 Results from Scenario V, sensor noise evaluation. Comparison of classification accuracy between Scenario IV and Scenario V. 198
7.15 Results from Scenario V, actuator noise evaluation. Comparison of classification accuracy between Scenario IV and Scenario V. 199
7.16 Results from Scenario V, actuator and sensor noise evaluation. Comparison of classification accuracy between Scenario IV and Scenario V. 201
7.17 Results from Scenario V, sensor permutations without occupancy. 205

List of Tables

7.2 Parameters for Scenario I, conflict function evaluation. 177
7.3 Parameters for Scenario II, pattern and step size evaluation. 179
7.4 Results for Scenario II, pattern and step size evaluation utilizing Δ_{NW}. 180
7.5 Results for Scenario II, pattern and step size evaluation utilizing Δ_{RBFN}. 181
7.6 Parameters for Scenario III. pattern and step size evaluation. 183
7.7 Results for Scenario III, pattern and step size evaluation. 184
7.8 Parameters for Scenario IV, non-noisy actuator and sensor evaluation. 186
7.9 Results for Scenario IV, non-noisy actuator and sensor evaluation. 187
7.10 Parameters for Scenario IV, sensor noise evaluation. 187
7.11 Results for Scenario IV, sensor noise evaluation. 188
7.12 Parameters for Scenario IV, actuator noise evaluation. 189
7.13 Results for Scenario IV, actuator noise evaluation. 189
7.14 Parameters for Scenario IV, actuator and sensor noise evaluation. 190
7.15 Results for Scenario IV, actuator and sensor noise evaluation. 190
7.16 Parameters for Scenario V, intermediate conflict calculation. 192
7.17 Results for Scenario V, intermediate conflict calculation. 193
7.18 Parameters for Scenario V, subset conflict calculation. 194
7.19 Results for Scenario V, subset conflict calculation. Intrinsic conflict (μ and σ) and classification accuracy. 195
7.20 Results for Scenario V, subset conflict calculation. Timing data. 196
7.21 Parameters for Scenario V, sensor noise evaluation. 197
7.22 Results for Scenario V, sensor noise evaluation. 197
7.23 Parameters for Scenario V, actuator noise evaluation. 198
7.24 Results for Scenario V, actuator noise evaluation. 199
7.25 Parameters for Scenario V, actuator and sensor noise evaluation. 200
7.26 Results for Scenario V, actuator and sensor noise evaluation. 201
7.27 Parameters for Scenario V, sensor permutations with occupancy. 202
7.28 Results for Scenario V, sensor permutations with occupancy. 203
7.29 Parameters for Scenario V, sensor permutations without occupancy. 204
7.30 Results for Scenario V, sensor permutations without occupancy. 206
7.31 Results for Scenario V, sensor permutations without occupancy. Sensor-specific classification impact. 207
7.32 Example results from Scenario VI, non-noisy optimization parameterized with SO(n). 210

7.33 Parameters for Scenario VI, non-noisy optimization parameterized with SO(n). 212
7.34 Results from Scenario VI, non-noisy optimization parameterized with SO(n) on hypothesis set $H_3^{\oplus}SE(2)$. 213
7.35 Results from Scenario VI, non-noisy optimization parameterized with SO(n) on hypothesis set $H_3^{\ominus}SE(2)$. 213
7.36 Parameters for Scenario VI, non-noisy optimization parameterized with GL(n,R). 214
7.37 Results from Scenario VI, non-noisy optimization parameterized with GL(n,R) on hypothesis set $H_3^{\oplus}SE(2)$. 214
7.38 Results from Scenario VI, non-noisy optimization parameterized with GL(n,R) on hypothesis set $H_3^{\ominus}SE(2)$. 215
7.39 Parameters for Scenario VI, noisy optimization parameterized with SO(n). . . . 215
7.40 Results from Scenario VI, noisy optimization parameterized with SO(n) on hypothesis set $H_3^{\oplus}SE(2)$. 217
7.41 Results from Scenario VI, noisy optimization parameterized with SO(n) on hypothesis set $H_3^{\ominus}SE(2)$. 218

List of Algorithms

1 Generic agent loop. 5
2 Intrinsic agent loop. 76
3 Initializing the sensorimotor graph. 110
4 Updating the sensorimotor graph. 112
5 Initializing the sensorimotor map. 117
6 Updating the sensorimotor map (1/2). 119
7 Updating the sensorimotor map (2/2). 120
8 Rebuilding a sensorimotor map with new manifold and projection matrices. . . 121
9 Initializing the bootstrapping classifier. 139
10 Updating the bootstrapping classifier. 139
11 Utilizing the bootstrapping classifier to identify the hypothesis with the minimal intrinsic conflict from a given set of hypotheses. 140
12 Initializing the bootstrapping particle swarm optimization. 143
13 Updating the bootstrapping particle swarm optimization. 144
14 Utilizing the bootstrapping particle swarm optimization to find a parameterization for the sensorimotor map that minimizes the intrinsic conflict. 146
15 Bootstrapping agent loop, utilizing the classifier approach. 150
16 Bootstrapping agent loop, utilizing the particle swarm optimization approach. . 151

Notation

General Notation

Symbol	Description
$\in$	set membership, where $a \in A$ denotes a being an element of the set A
$A = \{a \mid \cdots\}$	set builder notation for the set A, consisting of all a for which $\cdots$ holds
$\lvert A \rvert$	cardinality of the set A
$\mathbb{N}_0$	the set of natural numbers including zero, with $\mathbb{N}_0 = \{0, 1, 2, 3, \dots\}$
$\mathbb{N}^*$	the set of natural numbers excluding zero, with $\mathbb{N}^* = \{1, 2, ...\}$
$[x_1, x_2, \dots, x_n]$	row vector with n elements, where $n \in \mathbb{N}^*$
$[x_1, x_2, \dots, x_n]^T$	transposed row vector, i.e., column vector
0_n	zero column vector with n elements
$I_{m \times n}$	identity matrix with n rows and n columns
$0_{m \times n}$	zero matrix with m rows and n columns
$\mathbb{R}$	the set of real numbers with elements $r \in \mathbb{R}$
$\mathbb{R}^+$	positive real numbers including zero as $\mathbb{R}^+ = \{\, r \in \mathbb{R} \mid r \geq 0 \,\}$
$\mathbb{R}^n$	Euclidean space with dimension n
$\lVert x \rVert$	norm of the vector $x \in \mathbb{R}^n$, defined as $\lVert x \rVert = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}$
dim	dimension of the Euclidean n-space, thus $\dim(\mathbb{R}^n) = n$
$A \times B$	Cartesian product of A and B, as the set of ordered tuples (a, b) with $a \in A$ and $b \in B$
$f : A \to B$	function mapping elements $a \in A$ to elements $b \in B$
$g \circ f$	function composition of $f : A \to B$ and $g : B \to C$ as $g \circ f : A \to C$
$(a_1, a_2, \dots, a_n)$	sequence of length n
$\oplus$	append operation for sequences with $(a_1, a_2) \oplus a_3 = (a_1, a_2, a_3)$
$\mathcal{N}(\mu, \Sigma)$	multivariate normal distribution with mean μ and covariance Σ

Notation for Chapter 3 - Mathematical Foundations

Symbol	Description	Reference
$\mathcal{Q}^n$	n-dimensional configuration space with $q \in \mathcal{Q}^n$ and $\mathcal{Q}^n \subseteq \mathbb{R}^n$	Eq. (3.1)
DoF	degree of freedom	Section 3.2
ϱ	entity-specific occupancy function $\varrho : \mathcal{Q}^n \to \mathcal{B}$	Eq. (3.10)
$\mathcal{B}$	boundary set $\mathcal{B} = \{\beta_\circ, \beta_\bullet\}$	Eq. (3.11)
$\mathcal{Q}^n_\circ$ and $\mathcal{Q}^n_\bullet$	empty and occupied configuration space with $\mathcal{Q}^n_\circ \cup \mathcal{Q}^n_\bullet = \mathcal{Q}^n$	Eq. (3.12)
M	manifold	Section 3.3
(U_α, ϕ_α)	chart on manifold M with neighborhood $U_\alpha \subseteq M$ and coordinate function $\phi_\alpha : U_\alpha \to \mathbb{R}^n$	Eq. (3.14)
$\gamma_{(q_i,q_j)}$	path between the configurations $q_i, q_j \in \mathcal{Q}^n$ as the smooth function $\gamma_{(q_i,q_j)} : [0,1] \to \mathcal{Q}^n$	Eq. (3.17)
$\Gamma_{\mathcal{Q}^n}$	set of all paths on $\mathcal{Q}^n$	Eq. (3.19)
$\text{free}_\circ$	assesses, whether a path $\gamma_{(q_i,q_j)}$ on $\mathcal{Q}^n$ is unobstructed or obstructed as $\text{free}_\circ : \Gamma_{\mathcal{Q}^n} \to \{\mathit{true}, \mathit{false}\}$	Eq. (3.21)
$\text{free}_\otimes$	assesses the length of the unobstructed portion of a path $\gamma_{(q_i,q_j)}$ on $\mathcal{Q}^n$ as $\text{free}_\otimes : \Gamma_{\mathcal{Q}^n} \to [0,1]$	Eq. (3.24)
$t^\otimes$	return value of $\text{free}_\otimes$, with $t^\otimes \in [0,1]$ that denotes the fraction of the path $\gamma_{(q_i,q_j)}$ that is unobstructed	Eq. (3.25)
$\boxplus$	boxplus operator as the function $\boxplus : \mathcal{Q}^n \times \mathbb{R}^n \to \mathcal{Q}^n$, formalizing configuration changes on $\mathcal{Q}^n$ as addition of a difference $d \in \mathbb{R}^n$	Eq. (3.27)
$\boxminus$	boxminus operator as the function $\boxminus : \mathcal{Q}^n \times \mathcal{Q}^n \to \mathbb{R}^n$, that calculates the difference $d \in \mathbb{R}^n$ between two elements $q_i, q_j \in \mathcal{Q}^n$	Eq. (3.28)
$T_q\mathcal{Q}^n$	tangent space of the manifold $\mathcal{Q}^n$ as the set of all tangent vectors at $q \in \mathcal{Q}^n$	Section 3.4
$T\mathcal{Q}^n$	tangent bundle as the disjoint union of all tangent spaces of the manifold $\mathcal{Q}^n$	Eq. (3.35)
$\mathcal{X}$	vector field as distinct mapping $\mathcal{X} : \mathcal{Q}^n \to T\mathcal{Q}^n$ that assigns to every $q \in \mathcal{Q}^n$ a tangent vector from its tangent space $T_q\mathcal{Q}^n$	Eq. (3.36)
$\otimes$	metric tensor as the function $\otimes : T_q\mathcal{Q}^n \times T_q\mathcal{Q}^n \to \mathbb{R}$, assigning to a pair of tangent vectors a real number	Eq. (3.37)
length	assesses the length of a given path $\gamma_{(q_i,q_j)} \in \Gamma_\mathcal{Q}$ as a positive real number from $\mathbb{R}^+$	Eq. (3.38)
$\vec{\gamma}_{(q_i,q_j)}$	geodesic between $q_i, q_j \in \mathcal{Q}^n$ as a locally shortest path on $\mathcal{Q}^n$	Section 3.4
$\boxplus_{[T_q\mathcal{Q}^n]}$	boxplus operator as the function $\boxplus_{[T_q\mathcal{Q}^n]} : \mathcal{Q}^n \times T_q\mathcal{Q}^n \to \mathcal{Q}^n$, explicitly relating changes on $\mathcal{Q}^n$ to tangent space elements	Eq. (3.42)
$\boxminus_{[T_q\mathcal{Q}^n]}$	boxminus operator as the function $\boxminus_{[T_q\mathcal{Q}^n]} : \mathcal{Q}^n \times \mathcal{Q}^n \to T_q\mathcal{Q}^n$, explicitly relating distances on $\mathcal{Q}^n$ to tangent space elements	Eq. (3.43)
G	group as tuple of the set G and the group operation $\bullet$	Section 3.5
$\bullet$	group operation given as $\bullet : G \times G \to G$	Eq. (3.44)
id	identity group element as neutral element with respect to the group operation $\bullet$	Section 3.5

$GL(n, \mathbb{R})$	general linear group of dimension n as the set of all invertible $n \times n$ matrices with matrix multiplication as group operation	Eq. (3.50)
$SE(2)$	special Euclidean group of dimension 3. Element consists of 3×3 matrices, describing position and orientation in $\mathbb{R}^2$	Section 3.5
$SE(3)$	special Euclidean group of dimension 6. Elements consist of 4×4 matrices, describing position and orientation in $\mathbb{R}^3$	Section 3.5
$SE(n)$	special Euclidean group describing rotation and translation in $\mathbb{R}^n$	Section 3.5
$SE(2)$	special orthogonal group of dimension 1. Elements consist of 2×2 matrices describing orientation in $\mathbb{R}^2$	Section 3.5
$SO(3)$	special orthogonal group of dimension 3. Elements consist of 3×3 matrices describing orientation in $\mathbb{R}^3$	Section 3.5
$SO(n)$	special orthogonal group describing rotation in $\mathbb{R}^n$	Section 3.5
l_q	left translation as the function $\mathcal{Q}^n \rightarrow \mathcal{Q}^n$, which for matrix Lie groups corresponds to $l_q(x) = q \bullet x$	Eq. (3.52)
r_q	right translation as the function $\mathcal{Q}^n \rightarrow \mathcal{Q}^n$, which for matrix Lie groups corresponds to $r_q(x) = x \bullet q$	Eq. (3.53)
$\mathfrak{Q}^n$	Lie algebra as the tangent space of the Lie group $\mathcal{Q}^n$ with elements $\mathfrak{q}$	Section 3.6
$\hat{\mathfrak{g}}_1, \cdots, \hat{\mathfrak{g}}_n$	generators of the Lie group $\mathcal{Q}^n$ as basis vectors of its algebra $\mathfrak{Q}^n$	Section 3.6
$\mathfrak{q}$	element of the Lie algebra $\mathfrak{Q}^n$ as linear combination of its generators with $\mathfrak{q} = r_1\hat{\mathfrak{g}}_1 + r_2\hat{\mathfrak{g}}_2 + \cdots + r_n\hat{\mathfrak{g}}_n$ and $r_1, \cdots, r_n \in \mathbb{R}^n$	Eq. (3.56)
$[\cdot, \cdot]$	Lie bracket as the function $[\cdot, \cdot] \rightarrow \mathfrak{Q}^n \times \mathfrak{Q}^n \rightarrow \mathfrak{Q}^n$, that calculates the commutativity for two vector fields, given as $\mathfrak{q}_i, \mathfrak{q}_j \in \mathfrak{Q}^n$	Eq. (3.58)
exp	exponential map as the function $\exp : T_{id}\mathcal{Q}^n \rightarrow \mathcal{Q}^n$, that maps elements from the Lie algebra to the Lie group	Eq. (3.62)
log	inverse of the exponential map as the function $\log : \mathcal{Q}^n \rightarrow T_{id}\mathcal{Q}^n$, that maps elements from the Lie group to the Lie algebra	Eq. (3.63)
$\boxplus_{[\text{Lie}]}$	boxplus operator as the function $\boxplus_{[\text{Lie}]} : \mathcal{Q}^n \times T_{id}\mathcal{Q}^n \rightarrow \mathcal{Q}^n$, explicitly utilizing the matrix exponential	Eq. (3.66)
$\boxminus_{[\text{Lie}]}$	boxminus operator as the function $\boxminus_{[\text{Lie}]} : \mathcal{Q}^n \times \mathcal{Q}^n \rightarrow T_{id}\mathcal{Q}^n$, explicitly utilizing the matrix logarithm	Eq. (3.67)
τ	linear map $\tau : \mathbb{R}^m \rightarrow T_{id}\mathcal{Q}^n$ that associates controls $c \in \mathbb{R}^m$ to tangent space elements from $T_{id}\mathcal{Q}^n$	Eq. (3.70)

Notation for Chapter 4 - Sensorimotor Agents

Symbol	Description	Reference
$S_{(\Leftrightarrow)}$	sensorimotor system as tuple $(\mathcal{Q}^n, \varrho, C_\odot, \mathcal{S}, \alpha, \Pi)$, specifying the environment, agent properties and the interaction between both	Eq. (4.1)
$\mathcal{A}$	entity-specific set of actuators with $\mathcal{A} = \{ a \}$	Eq. (4.2)
C	control space as m-dimensional Euclidean space $C \equiv \mathbb{R}^m$, with m corresponding to the DoF of the entity's actuator	Eq. (4.3)
C_*	extended control space as n-dimensional Euclidean space $C_* \equiv \mathbb{R}^n$, with n corresponding to the dimension of $\mathcal{Q}^n$	Eq. (4.4)
τ_*	mapping from C to C_* with $\tau_* : C \to C_*$	Eq. (4.5)
T_*	matrix for τ_* that maps vectors from C to C_*	Eq. (4.6)
valid	entity- and actuator-specific function that assesses whether a control $c \in C$ is valid and can be processed by the actuator	Eq. (4.7)
$C_\odot$	entity- and actuator-specific set of valid controls given as $C_\odot = \{ c \in C \mid \text{valid}(c) = \mathit{true} \}$	Eq. (4.8)
$0_\odot$	zero control $0_\odot \in C_\odot$	Section 4.3.2
$C_\circledast$	image of $C_\odot$ under τ_* as the entity- and actuator-specific set of valid controls, mapped into the extended control space C_*	Eq. (4.9)
$C_\circledcirc$	extended control space envelope as Euclidean n-ball in C_* with radius $\epsilon_\circledcirc \in \mathbb{R}^+$ for which holds that $C_\circledast \subseteq C_\circledcirc$	Eq. (4.10)
α	transition function with $\alpha : \mathcal{Q}^n \times C_\odot \to \mathcal{Q}^n$, that defines the relation between valid controls and configuration changes	Eq. (4.11)
$\tau_\cup$	mapping from the extended control space to the tangent space of $\mathcal{Q}^n$ as $\tau_\cup : C_* \to T_{id}\mathcal{Q}^n$	Eq. (4.12)
$T_\cup$	matrix for $\tau_\cup$ that maps vectors from C_* to $T_{id}\mathcal{Q}^n$	Eq. (4.13)
$\tau_\triangleright$	composition of exp and $\tau_\cup$ with $\tau_\triangleright : C_* \to \mathcal{Q}^n$, that maps elements from the extended control space to group elements	Eq. (4.14)
$\tau_\blacktriangleright$	composition of exp, $\tau_\cup$ and τ_* with $\tau_\blacktriangleright : C \to \mathcal{Q}^n$, that maps elements from the control space to group elements	Eq. (4.17)
ϵ_c	noise vector sampled from an agent-specific multivariate normal distribution as $\epsilon_c \sim \mathcal{N}(0_m, \Sigma^c_{m\times m})$	Eq. (4.20)
$\tilde{c}$	potentially noisy equivalent of a given control $c \in C_\odot$	Eq. (4.21)
path	function that relates valid controls to the set of paths on the configuration space, with $\text{path} : \mathcal{Q}^n \times C_\odot \to \Gamma_{\mathcal{Q}^n}$	Eq. (4.23)
$\alpha_\otimes$	partial transition function $\alpha_\otimes : \mathcal{Q}^n \times C_\odot \to \mathcal{Q}^n$ that accounts for obstructions in the configuration space	Eq. (4.31)
$\alpha^\triangleleft_\otimes$	partial transition function that returns feedback proportional to the unobstructed path in $\mathcal{Q}^n$ as $\alpha^\triangleleft_\otimes : \mathcal{Q}^n \times C_\odot \to \mathcal{Q}^n \times [0, 1]$	Eq. (4.33)
$\tau_\cup$	mapping from the tangent space of $\mathcal{Q}^n$ to the extended control space C_* as $\tau_\cup : T_{id}\mathcal{Q}^n \to C_*$	Eq. (4.36)
$T_\cup$	matrix for $\tau_\cup$, that maps vectors from $T_{id}\mathcal{Q}^n$ to C_*	Eq. (4.37)

$\tau_{\blacktriangleleft}$	composition of $\tau_{\circlearrowleft}$ and log with $\tau_{\blacktriangleleft} : \mathcal{Q}^n \to C_*$, that maps elements from the configuration space to the extended control space	Eq. (4.40)
$\mathcal{P}$	entity-specific j-tuple of sensors with $\mathcal{P} = (p_1, p_2, \ldots, p_j)$ with $j \in \mathbb{N}_0$, i.e., an entity not having a sensors is valid	Eq. (4.43)
$\mathcal{I}_i$	sensor-specific perception set with r_i-dimensional real vectors $\tilde{\varpi}_i \in \mathcal{I}_i$ where $\mathcal{I}_i \subseteq \mathbb{R}^{r_i}$ associated to the sensor p_i	Eq. (4.44)
$\mathcal{S}$	entity-specific sample set as the cartesian product of all $\mathcal{I}_i$ as $\mathcal{S} \subseteq \mathbb{R}^k$ with elements $s \in \mathcal{S}$ as k-dimensional real vectors	Eq. (4.45)
π_i	sensor-specific perception function as $\pi_i : \mathcal{Q}^n \to \mathcal{I}_i$, that defines the relation between $q \in \mathcal{Q}^n$ and perceptions $\varpi_i \in \mathcal{I}_i$	Eq. (4.48)
$\epsilon_{(p,i)}$	noise vector, sampled from a sensor-specific multivariate normal distribution as $\epsilon_{(p,i)} \sim \mathcal{N}(0_{r_i}, \Sigma^{p_i}_{r_i \times r_i})$	Eq. (4.49)
$\tilde{\pi}_i$	sensor-specific noisy perception function $\tilde{\pi}_i : \mathcal{Q}^n \to \mathcal{I}_i$ that yields noisy perceptions $\tilde{\varpi}_i \in \mathcal{I}_i$	Eq. (4.50)
Π	sample function $\Pi : \mathcal{Q}^n \to \mathcal{S}$, mapping configurations to j-tuples, composed of noisy perceptions $\tilde{\varpi}_i \in \mathcal{I}_i$ calculated by $\tilde{\pi}_i$	Eq. (4.53)
PERCEPT()	percept function modeling the interaction between the entity and the unknown environment based on Π	Eq. (4.56)
ACT()	act function modeling the interaction between the entity and the unknown environment based on $\alpha^{\triangleleft}_{\otimes}$	Eq. (4.57)
$c^{\ominus}$	scheduled control as an element from $C_{\odot}$	Eq. (4.58)
$c^{\otimes}$	effective control given as $c^{\otimes}_i = t^{\otimes}_i c^{\ominus}_i$ with the scaling factor $t^{\otimes}_i$ calculated as $t^{\otimes}_i = (\text{free}_{\otimes} \circ \text{path})(q_i, c^{\ominus}_i)$	Eq. (4.59)
INIT()	sensorimotor map initialization function with arguments $s_1, \mathcal{Q}^n, T_{\circlearrowleft}$ and $T_{\circlearrowleft}$	Eq. (4.61)
UPDATE()	sensorimotor map update function incorporating new information given as the tuple $(c^{\ominus}_{i-1}, c^{\otimes}_{i-1}, s_i)$	Eq. (4.62)
DELIBERATE()	sensorimotor map deliberation function returning the next control $c^{\ominus}_i$ to be executed	Eq. (4.63)

Notation for Chapter 5 - Sensorimotor Maps

Symbol	Description	Reference
L_t	sensorimotor sequence of length t given as the consecutive list $L_t = (s_1, (c_1^{\ominus}, c_1^{\otimes}), s_2, (c_2^{\ominus}, c_2^{\otimes}), \cdots, s_t)$, with $s_i \in \mathcal{S}$ and $c_i^{\ominus}, c_i^{\otimes} \in C_{\odot}$	Eq. (5.1)
l_i	sensorimotor link as 3-tuple $l_i = (s_i, [c_i^{\ominus}, c_i^{\otimes}], s_{i+1})$	Eq. (5.2)
ω_i	memory node created from elements in L_t that relates consecutive paths on Q^n to elements from $\mathcal{S}$ and $\mathcal{B}$	Eq. (5.3)
$q_{(\omega,i)}$	configuration from Q^n that is associated to the memory node ω_i	Eq. (5.12)
$s_{(\omega,i)}$	sample from $\mathcal{S}$ that is associated to the memory node ω_i as $s_{(\omega,i)} = \Pi(q_{(\omega,i)})$	Eq. (5.13)
Ω_t	sensorimotor chain, created from L_t as a sequence of t memory nodes given as $\Omega_t = \omega_1, \omega_2, \omega_3, \cdots, \omega_t$	Eq. (5.11)
$C_{(\Omega,\odot)}[\omega_i, \omega_j]$	sequence of effective controls, corresponding to the connected list of consecutive geodesics on Q^n that connects ω_i and ω_j	Eq. (5.14)
$\mathcal{R}_{\odot}$	motor cover as generic encoding of the relation between $C_{\odot}$ and Q^n as $\mathcal{R}_{\odot} = \{(c, q) \mid q = \tau_{\blacktriangleright}(c),\ c \in C_{\odot}\}$	Eq. (5.16)
$\mathcal{M}_{\odot}[q_i]$	motor neighborhood as the accessible patch around $q_i \in Q^n$ with $\mathcal{M}_{\odot}[q_i] = \{(c, q) \mid q = q_i \bullet q_j,\ (c, q_j) \in \mathcal{R}_{\odot}\}$	Eq. (5.17)
$\Lambda_{\odot}[q_i]$	sensorimotor neighborhood defining perceivable patches around $q_i \in Q^n$ which relate $c \in C_{\odot}$ to $s \in \mathcal{S}$ and $\beta \in \mathcal{B}$, respectively	Eq. (5.25)
$\Lambda_{\odot}[\omega_i]$	sensorimotor set of the memory node ω_i, given as the information associated to ω_i with respect to $s \in \mathcal{S}$ and $\beta \in \mathcal{B}$	Eq. (5.30)
$\mathcal{R}_{\circledcirc}$	extended motor cover as $\mathcal{R}_{\circledcirc} = \{(c, q) \mid q = \tau_{\triangleright}(c),\ c \in C_{\circledcirc}\}$	Eq. (5.32)
$\mathcal{M}_{\circledcirc}[q_i]$	extended motor neighborhood around $q_i \in Q^n$	Eq. (5.33)
$\Lambda_{\circledcirc}[q_i]$	extended sensorimotor neighborhood of $q_i \in Q^n$	Eq. (5.41)
$\Lambda_{\circledcirc}[\omega_i]$	extended sensorimotor set of the memory node ω_i	Eq. (5.46)
$\text{connected}_{\circledcirc}$	function that assesses whether two extended sensorimotor neighborhoods are connected by a control $c \in C_{\circledcirc}$	Eq. (5.54)
$\text{connected}_{\odot}$	function that assesses whether two extended sensorimotor neighborhoods are connected by executable controls $c \in C_{\odot}$	Eq. (5.54)
$\Lambda_{\circledcirc}[q_i \triangleleft q_j]$	reference frame transformation of elements from $\Lambda_{\circledcirc}[q_j]$ into the reference frame $\Lambda_{\circledcirc}[q_i]$	Eq. (5.66)
$\Lambda_{\circledcirc}[q_i \uplus q_j]$	reference frame transformation and projection of $\Lambda_{\circledcirc}[q_j]$ into $\Lambda_{\circledcirc}[q_i]$	Eq. (5.69)
$C_{(\Omega,\blacktriangleright)}[\omega_i, \omega_j]$	transformation from $q_{(\omega,i)}$ to $q_{(\omega,j)}$, created by applying $\tau_{\blacktriangleright}$ and $\bullet$ to effective controls from $C_{(\Omega,\odot)}[\omega_i, \omega_j]$	Eq. (5.76)
$\Lambda_{\circledcirc}[\omega_i \triangleleft \omega_j]$	reference frame transformation of tuples from $\Lambda_{\circledcirc}[\omega_j]$ into the reference frame at $\Lambda_{\circledcirc}[\omega_i]$	Eq. (5.91)

$\Lambda_{\circledcirc}[\omega_i \uplus \omega_j]$	reference frame transformation and projection of tuples from $\Lambda_{\circledcirc}[\omega_j]$ into the frame at $\Lambda_{\circledcirc}[\omega_i]$	Eq. (5.94)
$\Lambda_{\circledcirc}[\omega_i \uplus \Omega_t]$	sensorimotor projection set of the sensorimotor chain Ω_t into the reference frame of the memory node ω_i	Eq. (5.97)
$\lambda_{(\mathcal{S},j)}$	data point as element of $\Lambda_{(\mathcal{S},\circledcirc)}[\omega_i \uplus \Omega_t]$	Eq. (5.98)
$\lambda_{(\mathcal{B},j)}$	data point as element of $\Lambda_{(\mathcal{B},\circledcirc)}[\omega_i \uplus \Omega_t]$	Eq. (5.99)
$\Lambda[\Omega_t]$	sensorimotor graph as the set of all individual sensorimotor projection sets at the memory node $\omega_i \in \Omega_t$, given as $\Lambda[\Omega_t] = \{\, \Lambda_{\circledcirc}[\omega_i \uplus \Omega_t] \,\vert\, \omega_i \in \Omega_t \,\}$	Eq. (5.100)
Ξ_u	anchor graph as the set of u anchor nodes, given as $\Xi_u = \{\, \xi_1, \xi_2, \ldots, \xi_u \,\}$	Eq. (5.101)
$\omega_{(\xi,i)}$	memory node $\omega \in \Omega_t$ associated to $\xi_i \in \Xi_u$	Eq. (5.102)
$\Lambda_{\circledcirc}[\xi_i \uplus \Omega_t]$	sensorimotor projection set of the memory node $\omega_{(\xi,i)}$ the anchor node ξ_i is associated to	Eq. (5.109)
$\Lambda[\Xi_u]$	sensorimotor map as the set of all individual sensorimotor projection sets at $\xi_i \in \Xi_u$ with $\Lambda[\Xi_u] = \{\, \Lambda_{\circledcirc}[\xi_i \uplus \Omega_t] \,\vert\, \xi_i \in \Xi_u \,\}$	Eq. (5.110)
$d(c_i, c_j)$	distance between two controls $c_i, c_j \in C_*$ as $d : C_* \times C_* \to \mathbb{R}^+$	Eq. (5.111)

Notation for Chapter 6 - Bootstrapping

Symbol	Description	Reference
$Q^n_{\mathcal{H}}$	arbitrary, but fixed hypothesis Lie group with n being greater than or equal to the degree of freedom of the agent's actuator	Section 6.2.1
h_i	hypothesis as the tuple $h_i = (Q^n_{(\mathcal{H},i)}, T_{(\cup,i)}, T_{(\cup,i)})$ consisting of a hypothesis Lie group and the matrices mapping elements from C_* to $T_{id}Q^n_{(\mathcal{H},i)}$ and back	Eq. (6.1)
$\mathcal{H}_k$	set of k distinct hypotheses, given as the set $\mathcal{H}_k = \{ h_1, h_2, \ldots, h_k \}$	Eq. (6.2)
$\Upsilon[Q^n_{\mathcal{H}}]$	Lie group specific function mapping a vector of real numbers to the set of invertible $n \times n$ matrices with $\Upsilon[G^n_{Lie}] : \mathbb{R}^z \to GL(n, \mathbb{R})$	Eq. (6.3)
$\Upsilon[Q^n_{\mathcal{H}}]^{\oplus}$	Lie group specific function mapping a vector of $z = n^2$ real numbers to $GL(n, \mathbb{R})$ utilizing exp with $\Upsilon[Q^n_{\mathcal{H}}]^{\oplus} : \mathbb{R}^z \to GL(n, \mathbb{R})$	Eq. (6.4)
$\Upsilon[Q^n_{\mathcal{H}}]^{\ominus}$	Lie group specific function mapping a vector of $z = \binom{n}{2}$ real numbers to $SO(n)$ utilizing exp with $\Upsilon[Q^n_{\mathcal{H}}]^{\oplus} : \mathbb{R}^z \to SO(n)$	Eq. (6.6)
$(\Xi \circ \Upsilon)[\Omega_t, Q^n_{\mathcal{H}}]$	Lie group specific sensorimotor map depending on Ω_t and a vector from $\mathbb{R}^z$ as $(\Xi \circ \Upsilon)[\Omega_t, Q^n_{\mathcal{H}}] : \mathbb{R}^z \to \Lambda[\Xi_u]$	Eq. (6.9)
Δ	conflict function that assigns to a given graph a real number, corresponding to the intrinsic conflict, given as $\Delta : \Lambda[\Xi_u] \to \mathbb{R}^+$	Eq. (6.12)
$\Lambda^{\bullet}_{(\mathcal{S},i,p)}$	non-empty subset of the sample projection set of the anchor node $\xi_i \in \Xi_u$ as $\Lambda^{\bullet}_{(\mathcal{S},i,p)} \subset \Lambda_{(\circledcirc,\mathcal{S})}[\xi_i \uplus \Omega_t]$	Eq. (6.15)
$\Lambda^{\bullet}_{(\mathcal{B},i,p)}$	non-empty subset of the obstruction projection set of $\xi_i \in \Xi_u$ as $\Lambda^{\bullet}_{(\mathcal{B},i,p)} \subset \Lambda_{(\circledcirc,\mathcal{B})}[\xi_i \uplus \Omega_t]$	Eq. (6.16)
k	kernel function that assigns to elements from $C_{\circledcirc}$ a positive real number as $k : C_{\circledcirc} \times C_{\circledcirc} \to \mathbb{R}^+$	Eq. (6.21)
Δ_{NW}	conflict function based on the Nadaraya-Watson kernel estimator	Eq. (6.30)
Δ_{RBFN}	RBFN conflict function	Eq. (6.38)
$(\Delta \circ \Xi \circ \Upsilon)[\Omega_t, Q^n_{\mathcal{H}}]$	Lie group specific conflict function depending on Ω_t and a vector from $\mathbb{R}^z$ as $(\Delta \circ \Xi \circ \Upsilon)[\Omega_t, Q^n_{\mathcal{H}}] : \mathbb{R}^z \to \mathbb{R}^+$	Eq. (6.41)

1 Introduction

In many domains, autonomous systems are the key technology for tackling the challenges of complex scenarios. An autonomous system in this context is a system that has an objective and chooses its own actions in order to pursue this goal. Complex scenarios are especially those where the focus lies on decision making under uncertain and incomplete knowledge about the world and which require reacting to unforeseen events or under limited time constraints. With respect to the scenarios, especially those are relevant, where human interaction is currently still mandatory. Prominent examples for these scenarios can be identified in both the context of terrestrial applications and the exploration of outer space.

Within the former, the subject area of automated driving stands out as a challenging research topic with increasing relevance [Urmson and Whittaker, 2008]. The motivation is to increase traffic safety by providing driving assistance as well as to reduce the driver's interaction with the vehicle, allowing him to focus on other activities. Ultimately, the goal is to achieve full automation [SAE International, 2018]. Though systems that assist the driver in individual scenarios (e.g. parking, lane keeping, adaptive cruise control on highways) exist and can be considered state-of-the-art in many modern medium-sized vehicles, integrated systems capable of performing fully automated driving without human supervision and interaction in all traffic situations do not exist yet.

Within the latter, developing systems for unmanned space exploration becomes more and more important, as exploration missions to small bodies like asteroids and comets [Probst et al., 2015] or planetary exploration with the aim of searching for extraterrestrial life pose a significant challenge [Starek et al., 2016]. Here, a prime motivation is the fact that with increasing distances, teleoperation, i.e., controlling a spacecraft remotely, becomes impractical. As stated in [Bajracharya et al., 2008], the *"round-trip communication delay between Earth and Mars ranges from 8 to 42 minutes"*, which poses a problem especially in situations where the need for quick decision-making arises. Consequently, trying to manually control the descent of a

spacecraft and react to unforeseen events is futile. Though past as well as current missions are successfully tackling these problems by successively increasing the autonomous capabilities of systems operating in interplanetary space [Muscettola et al., 1998, Jonsson et al., 2000] or on the surface of Mars [Bajracharya et al., 2008, Welch et al., 2013], upcoming missions may require new approaches providing a higher amount of autonomy and enabling spacecrafts to autonomously react to the situations they encounter [Starek et al., 2016].

It has to be noted that with respect to technical systems, the terms 'automated' and 'autonomous' are often used synonymously as well as interchangeably. However, a distinction can be made with respect to the level of independence and the capability to learn. Where automated systems are solely utilizing pre-calculated behaviors, autonomous systems are able to adapt to changes and improve their capabilities. In this regard, [Russell and Norvig, 2003, p. 37] proposes that if an agent *"relies on the prior knowledge of its designer rather than on its own percepts, we say that the agent lacks autonomy"*. Thus, an automated heating control unit that monitors the temperature and switches the heating on and off depending on the measured room temperature or, consequently, advanced driver-assistance systems (ADAS) are more on the 'automated' side of the spectrum. Autonomous systems, on the other hand, are systems that perform reasoning about the world itself and are thus able to react more robustly to unforeseen events and disturbances, i.e., they are able to react to events that have not been foreseen to the full extent by the engineers that created them. The research on intelligent agents, i.e., systems that provide said level of autonomy, has been a continuous endeavor for decades, where [Russell and Norvig, 2003, Chapter 1] provides a comprehensive introduction to the topic and an overview of past developments.

1.1 Rational Agents

According to [Russell and Norvig, 2003, p. 32], an agent is *"anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators"*, where [Wooldridge, 2000, p. 2] considers *"agents to be systems that are situated or embodied in some environment"* and elaborates on the quality of this interaction, stating that agents *"are not merely observers of their environment, nor are they the passive recipient of actions performed by other entities. Rather, agents are the active, purposeful originators of action"* [Wooldridge, 2000, p. 1]. Though both definitions make no initial assumptions on the nature of the environment or the nature of the interaction, they imply a certain degree of embodiment - not necessarily in the physical sense, but in the notion that a distinction can be made between the 'environment' and the 'agent'. With this separation of 'outside' and 'inside' in mind, sensors and actuators define the interface between these two. With respect to this definition, a sensor is a component associated to the agent that allows it to passively perceive information about the environment, where consequently an actuator is a component that allows the agent to actively change it. This very broad definition does not impose any constraints to the type of sensors, actuators, or environments. Though robotic agents, i.e., systems moving through the physical world, are often associated with the term agent, software agents are also valid instances of the agent paradigm. With respect to these types of agents and their differences, [Russell and Norvig, 2003, p. 32] states that a *"robotic agent might have cameras and infrared range sensors for sensors and*

various motors for actuators", where a software agent *"receives keystrokes, file contents, and network packages as sensory inputs and acts on the environment by displaying on the screen, writing files, and sending network packets"*. Within the context of this thesis, we consider the former definition and focus on simulated robotic agents, i.e., we consider scenarios in which an agent is situated in a simulated spatial environment where sensors allow it to perceive information available at the current location, and motor controls allow the agent to move from one location to another. As technically sensory inputs could capture properties not only associated to the environment but to the agent itself, a clarification with respect to the nature of 'inside' and 'outside' has to be made. Following the definition proposed by [Russell and Norvig, 2003, p. 44], we distinguish between the 'agent program' that defines the algorithmic component able to process sensory and motor information and the 'agent architecture', which corresponds to the hardware. This is in line with the interpretation given by [Pierce and Kuipers, 1997], where in the context of learning the agent architecture is described as *"a machine (physical or simulated) that the learning agent must learn how to use"*. In the remainder of this thesis, we utilize the term 'agent' as corresponding to the 'agent program' definition, while, unless otherwise stated, we don't explicitly consider a fixed 'agent architecture'. We correspondingly denote the 'state' or 'configuration' of the agent as the information required to describe the agent's pose (position and orientation) in the environment, where sensor information captures certain sensor-specific properties of said state.

The agent definition given thus far only covers the interaction between the agent and its environment, but does not make any statement with respect to the agent's behavior. A central concept in this regard is that of 'rationality', where [Russell and Norvig, 2003, p. 1] defines that a *"system is rational if it does the 'right thing', given what it knows"*. With respect to rational agents, [Russell and Norvig, 2003, p. 4] states that a *"rational agent is one that acts so as to achieve the best outcome, or when there is uncertainty, the best expected outcome"*. Thus, in addition to the sensors and actuators that allow the agent to interact with its environment, the agent needs a component that accounts for said rational behavior, where we denote this element as the 'behavior' or 'decision making' component. Thus, a rational agent can be seen as an entity composed of three building blocks: a sensor component, an actuator component (both often depicted as integrated in a single input/output-layer), and a decision making component, whereas this conceptual layout is depicted in Fig. 1.1, which is based on [Russell and Norvig, 2003, p. 33, Figure 2.1]. The subtleties of the behavior component are specific to the type of agent, where some examples are 'simple reflex agents', 'model-based reflex agents', 'goal-based agents', and 'utility-based agents', which are introduced and discussed in [Russell and Norvig, 2003, Chapter 2.4]. Within the context of this thesis, we mainly consider model-based agents. These kinds of agents maintain an internal representation of the world that allows the agent to *"keep track of the part of the world it can't see now"*, where this representation *"depends on the percept history and thereby reflects at least some of the unobserved aspects of the current state"* [Russell and Norvig, 2003, p. 48]. With respect to our scenario of an agent moving through a spatial environment, such a representation has to keep track of the agent's current position and orientation and allow the agent to integrate sensory inputs associated to states it has been in. Additionally it has to facilitate motion planning, i.e., allow the agent to determine the motor control that is required to move the agent from its current state to another one. In

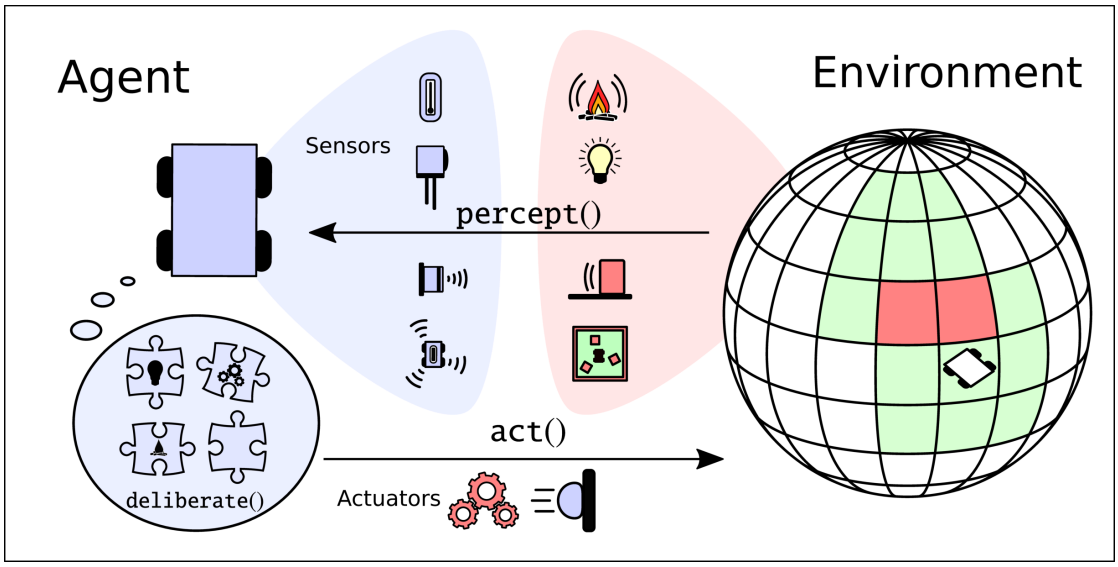


Figure 1.1: Conceptual layout of a rational agent. It perceives the environment it is situated in through various sensors and acts upon the environment through actuators. How information perceived by sensors is processed and what actuator commands are executed depends on the behavior component that deliberates on the next action and is specific to the agent and the domain. Figure based on [Russell and Norvig, 2003, p. 33, Figure 2.1].

the following, we denote this kind of representation as a 'map', where, at this point, we make no further assumptions with respect to its internal structure or additional properties. However, it is important to note that maps often can't be considered to be faithful representations of the environment as, stated by [Russell and Norvig, 2003, p. 462], *"agents almost never have access to the whole truth about their environment"*. Thus, maps have to be suitably chosen to match the requirements of the scenarios they are utilized in, providing means to integrate noisy sensor measurements or explicitly represent uncertain information. A more in-depth introduction to map representations, especially with respect to robot navigation, can be found in [Thrun et al., 2005].

Sensors, actuators, and the abstract map are utilized in the agent control loop. A generic example is given in Algorithm 1, which is based on the basic agent control loop defined in [Wooldridge, 2000, p. 26, Figure 2.1] and the concept of a model-based reflex agent as introduced in [Russell and Norvig, 2003, p. 49, Figure 2.11]. It depicts a cycle that is composed of a recurring sequence of three fundamental steps: perceiving information, reasoning about the next action, and executing it. In the perception step, the agent utilizes its sensors to perceive information associated with the current state of the environment. It then integrates these sensor measurements into the map, updating it with new information. In the reasoning step, the agent deliberates on its goals and selects an appropriate next action, where, in the act step, this action is executed by utilizing its actuators.

Though this generic agent loop captures the fundamental concept, there exist a vast number of different approaches with respect to architectures, communication standards, planning, and

Generic Agent Loop

```
1: procedure AgentLoop
   ▷ Initialize internal representation
2:     map ← Init()
       ▷ Execute main loop as long as the agent is active
3:     while Active() do
       ▷ Percieve environment
4:         map ← Percept()
           ▷ Update internal representation
5:         map ← Update(perception)
           ▷ Deliberate on next action
6:         action ← Deliberate(map)
           ▷ Invoke actuator to perform state transition
7:         Act(action)
8:     end while
9: end procedure
```

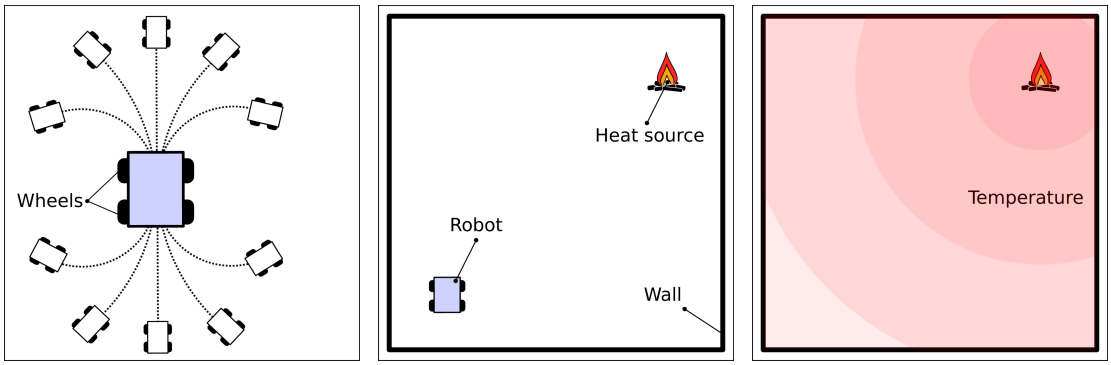
Algorithm 1: Generic agent loop. Implemented as a repeating sequence of perceiving, deliberating and acting. Based on the basic agent control loop depicted in [Wooldridge, 2000, p. 26, Figure 2.1] and the concept of a model-based reflex agent as introduced in [Russell and Norvig, 2003, p. 49, Figure 2.11].

coordination that would exceed the scope of this thesis. A good introduction to rational agents can be found in [Russell and Norvig, 2003, Wooldridge, 2000], while a survey on current decision making strategies and multi-agent systems can be found in [Balke and Gilbert, 2014] and [Dorri et al., 2018], respectively.

1.2 Robot Navigation

As previously stated, rational agents are a paradigm that can be utilized to approach a vast number of tasks from different virtual or non-virtual domains. However, in many challenging scenarios, agents are situated in some kind of ambient space endowed with a certain topological structure in which they have to move in order to achieve their goal. Examples for such domains include automated driving [Urmson and Whittaker, 2008], underwater robotics [Chutia et al., 2017], aerial robotics [Feron and Johnson, 2008], space exploration [Starek et al., 2016], and domestic robotics [Prassler and Kosuge, 2008]. While in some scenarios the ambient space is considered to be two-dimensional, i.e., it corresponds to the two-dimensional Euclidean space, in many domains robots have to navigate through three-dimensional environments. In all of these domains, a pivotal task is to monitor and control the position and orientation while simultaneously satisfying certain boundary conditions (e.g. speed limit, obstacle avoidance).

As established in Section 1.1, agents interact with their environment by utilizing sensors and actuators, where sensors allow the agent to perceive information and properties of the environment like temperature, the distance to an obstacle, a camera image, or the sound intensity. Motor controls have a geometric interpretation, allowing the agent to manipulate its pose, i.e.,



(a) Top-down view of the four-wheeled robot system equipped with a simple temperature sensor. Schematic depiction of the effect of all ten different actuator commands.

(b) Top-down view of enclosed environment with robot starting location and a heat source.

(c) Schematic depiction of the measurable temperature at each (x, y) position. Color intensity corresponds to temperature. Discretization of the continuous temperature gradient for illustrative purpose only.

Figure 1.2: Layout of a simple robotic system and the environment it is situated in.

its position and orientation, with respect to its environment. Within these types of scenarios, state representation, deliberative capability, and task complexity are tightly connected, as more complex tasks require more complex decision-making capabilities, which in turn require more complex state representations.

1.3 Motivational Example

As a motivational example, we consider the simple robot depicted in Fig. 1.2a. The robot has four wheels that can be controlled such as to allow it to move forward, backward, and on circular paths with different radii. For illustrative purposes, we allow the robot to only execute a fixed set of ten motor controls with given turn radius and travel distance that are depicted in Fig. 1.2a. Thus, the robot can control its position and orientation. It is also endowed with a temperature sensor, allowing it to measure the ambient temperature of the environment at the location it currently is at. The interaction between the robot and its environment is modeled to be sequential, i.e., the robot can either perceive sensor information or execute a motor control. The robot is situated in a two-dimensional environment that is enclosed by walls, where a top-down view is given in Fig. 1.2b. The environment is empty, except for a heat source that causes a radial monotonous decreasing heat distribution in the environment, which is illustrated in Fig. 1.2c, where for illustrative purposes we have discretized the otherwise continuous temperature gradient. For this example, we assume both measurements and controls to be noise-free and for the sake of brevity omit any hazardous effects collisions with the walls might have on the robot. We now challenge the robot with tasks of varying complexity and discuss the impact these tasks have on the behavior and ultimately on the required state representation.

Task I: Within the first task, the robot starts at an arbitrary location with the goal to use its means of locomotion to reach the heat source. Based on the means of perceiving the environment, this corresponds to the task of reaching the location where the measured temperature is the highest. A simple approach is to measure the temperature, apply a random motor control, perform a second temperature measurement at the new location, and compare these two measurements. If the second temperature measurement is higher than the first, the robot has moved towards the heat source. Thus, the robot has reached a new best state and it repeats the aforementioned steps. However, if the second measured temperature is lower, the robot actually moved away from the heat source. Thus the current state is less desirable and the robot has to move back to where it started by reversing the last motor command. After re-entering the previous best state, the sequence starts again, i.e., a new random motor control is generated, executed, and the new position is evaluated. To avoid the robot continuing to search indefinitely, we assume that after testing a certain number of adjacent states and finding none with a higher temperature, the algorithm terminates. While this strategy is not very efficient, as unnecessarily many controls are executed, it will nevertheless allow the robot to eventually reach its destination. An example path is depicted in Fig. 1.3a. This strategy can be compared to the first-choice hill-climbing algorithm [Russell and Norvig, 2003, p. 113], where the difference is that in the mentioned algorithm, evaluating a new location precedes the state transition, where in our case the robot has to actually move to a new location in order to measure the temperature and evaluate this new state.[1] In this case, the robot does not require utilizing a designated representation of either its own state, or the means it has to change it. It has to remember the last two temperature measurements and the last executed control, the former to perform a comparison, the latter to be able to reverse the action, in case of the robot unintentionally having moved away from the heat source. However, no more information on either the nature of the sensor inputs or the controls has to be provided. This means that the information on what a temperature sensor is, i.e., the nature of the property it measures, is not relevant to the behavior. Additionally, the information on what the controls actually do is not relevant as the behavior utilizes only local properties of the environment and is does not model the effect of what actuator controls actually do.

Task II: We now switch the robot off, put it at the previous starting location, and switch it on again. The objective remains the same, namely for the robot to reach the heat source. As the robot has already located the heat source, a naive strategy would be to just repeat the sequence of actions it has performed before. However, this sequence would include redundant controls, namely controls that had to be reverted. A better strategy therefore is to initially clear the sequence of these kind of controls and only execute the modified and shorter sequence which is depicted in Fig. 1.3b. To be able to do so, the requirement has increased to having an extended memory. Not only has the robot to store the last perception, but the complete sequence of controls. Marking controls as reverted is required to be

[1] It has to be noted, that this algorithm faces the same problems as the first-choice hill-climbing algorithm as it is susceptible to getting stuck in local maxima or plateaus. However, in this case we know that the temperature is monotonically increasing, does not exhibit a plateau, and has only one global maximum, therefore this approach is applicable here.

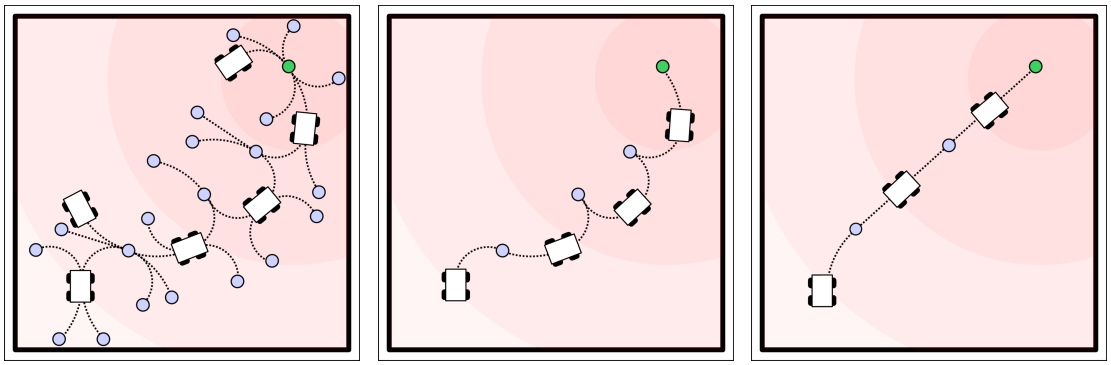
able to repeat the sequence and to exclude superfluous controls. Remarkably, even in this scenario, no requirement on the modeling of sensor information, controls, or topology is required.

Task III: We again switch the robot off, move it to the starting location and switch it on again. This time we change its objective so that it has to move to the heat source on the shortest path possible. It is obvious that the previous strategy of simply repeating the controls (even when excluding the superfluous ones) will not provide a valid solution.[2] Thus, the robot has to determine the location of the heat source relative to its starting location in order to be able to calculate a path to it based on the information it has collected during the exploration of the environment in Task I. It then has to find a suitable sequence of controls that allows it to reach the destination, i.e., from the set of all sequences of controls that reach the destination it has to find the one with the minimal distance traveled. In this task, the robot has to utilize a more complex internal representation of the environment and itself. Firstly, it has to establish a map, i.e., a reference frame of the environment that allows it to associate previously perceived sensor measurements with the locations they have been measured at. Secondly, it has to be able to express the effect of motor controls with respect to said map in order to plan a sequence of motor controls to traverse from its starting location to its destination.[3] Thus, to solve the problem, the agent has to assign a geometric interpretation to the environment and to its actions, i.e., it has to internally model the relation between both, based on previous experiences. Using this knowledge, the agent is able to calculate a direct path, i.e., a shortcut from its starting location to its destination, which is depicted in Fig. 1.3c.

Though all these scenarios required some kind of internal state representation, we have seen that the complexity of the different tasks has a direct impact on the required complexity of the state representation. In the first scenario, keeping track of past perceptions in order to perform comparisons and past actions to revert them is sufficient. The second scenario extends this requirement, as keeping track of a longer history of actions is required. However, only in the third scenario we actually have to provide information on the meaning of actions, i.e., the agent has to have some kind of interpretation about what its environment looks like with respect to its geometry. In addition, the same is required in terms of the system's motor controls, i.e., how turning the wheels has an impact on the robot's position and orientation. As demonstrated, simple agent behavior can be modeled in a very reactive fashion, where, based on some sensor perceived sensor inputs, the agent would perform some kind of immediate motor responses. More complex scenarios, however, would require more sophisticated behavior control approaches, potentially requiring mechanisms for maintaining a history of past interactions and an internal representation of the environment that models the effects that actions have on it.

[2] Though it is possible that a random sequence of controls coincides with the shortest sequence of controls, it is pretty unlikely.

[3] For the sake of brevity, we simply assume that the robot is able to create such a plan. An introduction to planning can be found in [Russell and Norvig, 2003, Chapter 11 and 12].



(a) Path from Task I as a result of stochastic hill climbing.

(b) Path from Task II as optimized path from Taks I with superfluous controls removed.

(c) Path from Task III, calculated as a shortcut based on knowledge on geometric relations.

Figure 1.3: Paths the robot traveled in Task I, Task II and Task III. Blue dots and robot images correspond to distinct positions the robot has been. The green dot corresponds to the location the maximal temperature has been measured. A dotted line indicates the path the robot has traveled between two distinct positions.

1.4 Motivation

As the examples have shown, the state representation a rational agent has of its environment, has to be tailored to both the domain as well as the designated task. As within our example, agents moving through an ambient space, are often tasked with the objective of moving to a certain location, which boils down to the goal of moving from a certain starting pose to a destination. Thus, with respect to this task of navigation, algorithms and representations are required that allow these kinds of agents to ask the questions *"where am I?"*, *"where am I going?"*, and *"how should I get there"* [Leonard and Durrant-Whyte, 1991]. The first question is associated to the tasks of localization and mapping, where mapping describes the task of establishing a representation of the environment that can serve as a frame of reference and localization corresponds to the closely related task of determining where the agent currently is with respect to said map. Though both tasks can be approached individually, in many cases both localization and mapping have to be performed simultaneously, a scenario accordingly denoted as simultaneous localization and mapping (SLAM) (see Section 2.4.1). The other two questions correspond to the tasks of reasoning, path planning, and control. Reasoning within this context denotes the capability to select a location the agent wants to move to. Path planning and control describe the tasks of computing a sequence of motor controls, that will allow the agent to move from its current location to the destination identified in the previous step as well executing this sequence, where, depending on the scenario, correcting and modifying this plan on the go might be necessary. As a foundation for answering all three questions, suitable map representations and models of the agent's motor capabilities have to be available.

1.4.1 Representations for Technical Systems

Within the robotics domain, various types of map representations [Thrun, 2002], many approaches for localization and mapping [Frese et al., 2010, Clemens, 2017], and many different methods for coping with uncertainty and noise [Lang et al., 2007, Ramos and Ott, 2016, Grimme et al., 2017] exist. However, these representations and algorithms often rely heavily on a priori knowledge, that has to be provided to the system in advance in terms of sensor and motion models. Conceptually, they model the relation between measurements, controls, and the map representation mathematically, which often requires to explicitly provide a geometric interpretation of both incoming (sensor measurements) and outgoing (motor controls) data. For example, utilizing LiDAR measurements for localization and mapping requires that a corresponding model has to be provided that allows to interpret raw measurements as angles and distances within the robot's own frame of reference, ultimately allowing to infer the position of obstacles in the environment. A model for an ultrasound sensor, however, would have to encode the relations between signal run-time, carrier medium density, and the distance (see [Thrun et al., 2005, Chapter 6]). The same holds for motion models as they have to model the relation between controls and the pose. A wheeled robot has to have a model that describes the effect motor controls have on its position and orientation in the plane, where a spacecraft or an underwater vehicle has to model the effects of its thrusters with respect to its position and orientation in a three-dimensional environment (see [Thrun et al., 2005, Chapter 5]). Additionally, to be able to effectively enable motion planning, these position and orientation changes have to be compatible with the map representation of the environment. An additional factor these models have to account for is noise, which holds true especially in real-world scenarios where having only imperfect and incomplete knowledge about the environment is more often rule than exception (see [Thrun et al., 2005, Chapter 1]). Having to manually craft and tune these models is not unproblematic due to several reasons:

Genericity and homogeneity: As previously stated, sensor and motion models are crucial for all map-related computations. Erroneous sensor models gravely impair localization and mapping performance while erronous motion models have a similarly severe impact on path planning and locomotion capabilities. As these models are potentially specific to the components, the domain, the task, the map representation, and the algorithms used for localization, mapping, and path planning, they require very detailed knowledge about these factors and their potential interdependence. Thus, manually implementing these models can be both time-consuming and error-prone, where [Olsson et al., 2006] states that *"it is extremely hard to design a robotic system to use its sensors in an efficient manner in challenging realistic and changing environments or with more complex robots"*, especially as it is *"very hard as a human designer to understand the environment in which the robot is acting in from the robots perspective"*. This is in line with a more conceptual difficulty as most approaches require constant transformation of data, as neither sensor integration nor motor control synthesis is usually a native trait of map representations. This means that mechanisms have to account for these transformations, i.e., from measurements to obstacle information or from planned paths represented in an arbitrary map format to the actuator controls.

Adaptability and robustness: Though approaches for learning probabilistic models for mobile robots exist [Eliazar and Parr, 2004, Visatemongkolchai and Zhang, 2007], sensor and actuator models are usually non-adaptive and thus attuned to the nominal functionality of the corresponding systems. Even noise parameters are usually assumed to be constant. However, both sensor and actuator parameters can change over the lifespan of a robotic system. This can happen either gradually as moving parts like joints and wheels are subject to wear and tear or abruptly due to mechanical or electronic failure. In addition to involuntary causes, parameters might deliberately change, too due to robot reconfiguration (e.g. different sensor placement, changed wheel diameter). Eventually, changed sensors and actuators require manually updating the corresponding models. This poses a challenge especially in scenarios where autonomous systems operate without the possibility of human intervention, face unknown or unexpected conditions, or have to explore environments sensor and actuator models are not tuned to. In these cases, localization and mapping performance is likely to be subpar, which in turn has a negative impact on deliberation and ultimately on the performance of the system as a whole.

1.4.2 Representations in Biological Systems

As introduced in Section 1.1, a rational agent is an entity that is capable of interacting with its environment and exhibits goal-directed behavior. However, this concept is not only applicable to technical, but also to certain biological entities, i.e., most animals. Both interaction with their environment and goal-directed behavior are fundamental traits of these organisms though naturally both deliberative capabilities as well as the complexity of the sensory and actoric systems vary tremendously. A common task, however, is the need to traverse their environment, e.g. in search for food or shelter. Thus, these organisms are challenged with the task of navigation, which, on a conceptual level, boils down to sensory integration, deliberation, and synthesis of motor controls. However, the individual structures responsible for processing of sensory inputs and motor outputs as well as those for decision-making are tightly connected and form integrated sensorimotor systems that exhibit interesting properties when it comes to genericity and homogeneity as well as adaptability and robustness:

Genericity and homogeneity: Opposite to the domain and task specific components that are required for modeling robotic system behavior discussed in the last section, the corresponding neural structures in most species are instances of a common architecture. This concept of a nervous system with afferent (sensory), efferent (motor), and central (integration and deliberative function) neural elements can be found in most organisms [Ghysen, 2003], where only a small set of animals (e.g. sponges) lack corresponding structures. Even in simple organisms such neural coupling between sensory and motor components can be observed [Jékely, 2011]. Though specialized structures have evolved to facilitate processing of different types of sensory input and control of different types of locomotion, the neuron is the common unit of information processing across all organisms. As perceiving sensor measurements, invoking motor controls, and deliberating is performed by a neuron based computational structure, there exist no architectural or structural break

with respect to atomic representation. Thus, we can observe a conceptual and structural genericity, i.e., a homogeneous structural integrated system.

Adaptability and robustness: The sensory and actoric systems of organisms and hence their sensorimotor capabilities are often subject to changes. Gradual and slow changes are more rule than exception, originating from natural causes as growth or aging. Both have to be accounted for by the neural structures responsible for information processing. Sudden changes can be caused by injuries, which may impair both sensory and actoric systems. For both types of changes compensatory mechanisms exist, where these adaption affects the whole sensorimotor system [Jarvis et al., 2013, Della-Maggiore et al., 2015]. As in real world scenarios, both sensory inputs and motor controls are inherently noisy, coping with these types of data obviously is a native trait [Faisal et al., 2008].

1.4.3 Biology as Inspiration

As apparently the nervous system provides effective means for navigation and decision-making in real-world environments, looking at it for inspiration when developing corresponding systems for robotic systems seems only natural. In this regard, two questions arise: The first one is whether properties as genericity, adaptability, and robustness have a cause in the structural layout, namely in the tight coupling of sensory, motor, and deliberative components. The second question is how information about geometric relations, transformations of reference frames, and the representation of the topology of the environment is represented, especially with respect to the genericity of the nervous system on the one hand, and the broad spectrum of locomotive capabilities and diversity of actuators on the other hand. To what extend these geometric concepts are genetically precoded and to what extend they are constructed through active exploration and learning is not yet answered. The plasticity of the intrinsic representation shown in several experiments suggests a dynamic and adaptable representation of geometric concepts.

1.5 Aim and Approach

As depicted in Section 1.3, a mobile entity situated in an ambient space, aiming on tackling more complex navigational tasks requires an internal representation that allows it to infer on geometric relations, i.e., the relation between the topology of the environment and its motor capabilities. Such a representation can be seen as composed of three integral components responsible for sensor integration, motor control synthesis, and environment representation, where in many technical systems these components correspond to a set of sensor and motion models and an underlying map representation. However, as discussed in Section 1.4.1, manually crafting and tuning these models and selecting a suitable map representation is a difficult task. This holds especially true in situations where information about the environment is scarce (making map selection and structure difficult), the input and output modalities of the system are subject to change (having an impact on sensor and motion models), or the overall behavior of the system changes due to changes in its environment (having an impact on the general interaction between these three components). However, in biological systems, as discussed in Section 1.4.2, sensor integration, motor control synthesis, and environment representation can be seen as integral

functions of an integrated sensorimotor system that exhibits a high degree of adaptability and robustness. This leads to the question whether technical systems could benefit by adopting these biological principles.

The aim of this thesis is to develop a method that allows a mobile entity to construct a consistent internal representation of the sensorimotor interaction with the environment it is situated in, utilizing only minimal a priori knowledge about its sensors, actuators, and the environment. A key concept of our approach is to assume a close coupling between sensory inputs and motor controls (see Sections 2.1.1 and 2.1.3), i.e., a sensorimotor-based representation of spatial environments. Within this thesis, we aim on combining ideas and findings from neurobiology, developmental psychology, navigational experiments, and the concepts of manifolds into a graph-based bootstrapping algorithm that enables a mobile entity to calibrate itself.

We decompose the problem into two tasks. The first one is to establish a suitable domain-independent and parameterizable representation of the entity's configuration space, i.e., the set of states the entity can be in. The second one is to utilize this representation for calibration, i.e., for adapting this representation to match the geometric relations between sensory inputs and motor controls.

1.5.1 Represention

With respect to the first task, we establish a manifold-based representation of the configuration space. It is fundamentally inspired by the view on the cognitive map discussed in [Kuipers, 1982, Kuipers, 1983], where the 'map in the head' is proposed to have the character of an 'atlas', a structure where local charts that represent distinct places with their local metric reference frames are connected by topological relations. We relate this metaphorical notion of an atlas to the concept of an atlas as defined in the context of manifolds (see Section 3.3). Based on this idea we construct a graph where each node corresponds to the origin of a distinct coordinate chart on the manifold expressed with respect to the entity's motor controls. Establishing a functional relation between motor controls and the configuration space as well as between the configuration space and sensory inputs allows us to represent local patches on the manifold with respect to their sensorimotor properties. Thus, the configuration space is represented as a set of overlapping sensorimotor neighborhoods anchored to a graph whose edges encode the control required to transition from one chart to another.

This allows us to draw parallels between the graph structure and the neurobiological concepts of representing places [O'Keefe and Nadel, 1978], where the major difference of our approach is that we utilize the graph only as a passive representation and not for localization, i.e., the nodes in our graph do not actively respond to sensory input at a given configuration, but solely encode spatial structure. Another similarity can be seen with respect to the sensorimotor environment given in [Buhrmann et al., 2013], which encodes the functional relation between motor and sensory input. The main difference lies in the locality of the graph structure that decomposes the sensorimotor environment into a set of independent neighborhoods.

This manifold-based graph structure can be related to the labeled graph that is proposed based on findings from navigational experiments discussed in [Warren et al., 2017] as it captures both the local metric properties but, due to the locality induced by the neighborhoods, would allow for incorporating non-Euclidean properties as well.

Utilizing Lie groups that correspond to rigid transformations in Euclidean space as the underlying manifold type relates the graph-based representation of the configuration space to the domain of holonomic and nonholonomic motion planning (see Section 2.3.4 and [Bloch et al., 2007]). A difference, however, is that most approaches utilize a predefined and fixed Lie group representation and use the group operation merely as a tool for representing known spatial relations. In this thesis, however, we utilize the Lie group representation as a parameterizable building block that facilitates a dynamic mapping from the set of controls to the configuration space. Thus, we utilize the Lie group to establish a 'parameterizable homeomorphism' that we try to calibrate based on the information available to the system.

1.5.2 Bootstrapping

Situated in the domain of bootstrapping (see Section 2.3.2) we aim on establishing a method for allowing a mobile entity to construct a consistent internal representation of the sensorimotor interaction with the environment it is situated in. In line with other approaches [Pierce and Kuipers, 1997, Philipona et al., 2004], we don't provide a priori information on the sensors or actuators to the system in advance, thus leaving both input and outputs uninterpreted. Therefore the task for the entity is to infer on the geometric relation between these two properties solely from information available from exploring the environment. Inspired by research from development psychology [Piaget, 1953], we utilize motor babbling (see Section 2.2.3) as an exploration strategy for establishing a dataset for building the representation from. Subsequently invoking sensors and actuators results in this dataset consisting of an alternating sequence of sensory inputs and motor controls that describes the subjective history of the entity's sensorimotor interaction with its environment.

The atomic elements this sensorimotor sequence consists of are comparable to the views and actions proposed in [Kuipers and Levitt, 1988], i.e., the sensorimotor features utilized in [Zetzsche et al., 2008] and [Nakath et al., 2014], both representing the change in sensory input with respect to motor actions. However, instead of attributing a fixed geometric interpretation to the motor component, we establish a parameterizable mapping between the agent's control space and the manifold-based representation of the environment. Utilizing Lie groups, a special kind of differentiable manifold (see Sections 2.3.3 and 2.3.4), allows us to parameterize the sensorimotor interaction with respect to the tangent space of the Lie group.

Adapting concepts from the field of molecular docking and protein folding (Section 2.3.5), we interpret the alternating sequence of sensory inputs and motor controls as a molecular-like flexible structure. Based on the interpretation proposed in [Mirzaei et al., 2015], we model this structure as being composed of rigid segments corresponding to the sensory inputs and flexible joints corresponding to the motor components. Assigning a geometric interpretation to the motor components induces a spatial structure to the whole sequence, i.e., choosing a certain parameterization 'folds' this sequence, causing the sensory components to have distinct relative alignments.

Based on the assumption, that the functional relation between motor controls and sensory inputs is locally smooth [Pierce and Kuipers, 1997, Philipona et al., 2003, Olsson et al., 2005, Olsson et al., 2006], we treat the task of modeling said functional relation as a nonparametric regression problem. This allows us to adapt the notion of the energy-based cost function used

in molecular docking and establishing a conflict measure that allows us to assign a degree of 'intrinsic consistency' or 'intrinsic conflict' to each neighborhood. The task of bootstrapping, i.e., finding a valid geometric interpretation of motor controls thus corresponds to finding a 'folding' that minimizes this conflict function for all neighborhoods.

We implement this algorithm in two distinct ways. The first implementation is a classification approach that detects the minimal-conflicting parameterization from a set of given hypotheses parameterizations. The second utilizes PSO (see Section 2.2.2) as a derivation-free optimization technique to find a valid parameterization based on a hypothesis of the underlying Lie group.

1.6 Research Questions and Contributions

We investigate the approach depicted in Section 1.5 with respect to the following questions:

- Is it possible to infer on the topology of the environment with respect to the interaction solely by processing a history of prior sensorimotor interactions?
- To what extend can we minimize the need for a priori knowledge for such a model?
- Is the local intrinsic conflict a reasonable measurement for the applicability of such parameterization?
- How robust is such a system with respect to noisy perceptions and motor controls?

To this end we conduct experiments in a virtual environment, thus we:

- Implement an extensible software framework consisting of the sensorimotor map representation, a simulation environment, and an agent framework for testing and evaluation of the approach.
- Evaluate the bootstrapping algorithm in various test cases with varying degrees of freedom in non-noisy simulated environments as well as evaluate the effect of noisy sensors and actuators.
- Discuss the developed algorithm with respect to both its strengths and limitations and elaborate on its extensibility.

The contributions of this thesis are as follows:

- Establishing a formal manifold-based mathematical foundation for representing sensorimotor systems, i.e., for representing the sensorimotor interaction of an entity with its environment.
- Defining the sensorimotor map, a domain-independent parameterizable graph-based sensorimotor representation with minimal requirement for incorporation of a priori knowledge.

- Development of intrinsic conflict measurements that allows to assess the local consistency of a given sensorimotor map.
- Combining the sensorimotor map and the conflict function which allows for classification and optimization algorithms to be utilized, ultimately proving an entity with means for bootstrapping.

1.7 Structure of the Thesis

The thesis is structured as follows:

- Chapter 2 gives a comprehensive overview over the biological and technical foundations and introduces relevant concepts that are utilized within this thesis.
- Chapter 3 provides an introduction of the pivotal mathematical concepts and the terminology that will be used throughout the thesis.
- Chapter 4 builds upon these concepts and relates them to the definition of a rational agent, i.e., providing a thorough formal foundation for sensory inputs, motor controls, and their relation to the configuration space.
- Chapter 5 covers the development of the sensorimotor map, a graph-based joint sensorimotor structure that establishes a tightly-coupled connection between the topology of the configuration space and the sensory and motor capabilities of the agent.
- Chapter 6 depicts the utilization of the sensorimotor map for assessing the topology of the configuration space, i.e., for detecting of the agent's sensorimotor properties.
- Chapter 7 briefly describes the implementation of the sensorimotor map as well as the simulation environment developed for the evaluation and covers the experimental setup, the execution of experiments, and a discussion of the results.
- Chapter 8 concludes the thesis by giving a summary over the work and highlighting further research questions.

At the beginning of each chapter, we give an overview of its structure as well as a synopsis of its contents.

2 State of the Art

In this chapter we give a thorough overview of the state of the art associated to the topic of this thesis and the concepts utilized within. As introduced in Chapter 1 the main subject of this thesis is the question on how, and to what degree, a mobile agent can infer on the properties of its sensorimotor interaction with its environment only provided with minimal a priori information on its sensory and motoric capabilities. We pursue this goal by joining concepts from biology and mathematics. With respect to biology we utilize a tight coupling between sensory and motoric capabilities. With respect to mathematics we utilize the concept of differentiable manifolds. This chapter aims to provide an overview of both the biological and technical foundations where the structure is as follows: Section 2.1 gives an introduction to the biological foundations both within the context of neurobiology and developmental psychology. Section 2.2 gives a brief overview of bio-inspired algorithms and technical systems and Section 2.3 introduces the key concepts this thesis is related to. More specifically in Section 2.3.1 we provide an overview of different types of map representations, Section 2.3.2 introduces the concept of bootstrapping, Sections 2.3.3 and 2.3.4 briefly introduce manifolds and Lie groups where Section 2.3.5 covers the topic of molecular docking and protein folding. Section 2.4 covers related topics as navigation in Section 2.4.1 and a brief overview of autonomous and bio-inspired robotics in Section 2.4.2.

2.1 Biology

In this section we present biological foundations from a fundamental perspective in Section 2.1.1 and focus on the regions of the brain associated with spatial representation and processing in Section 2.1.2. We will briefly discuss integrative properties in Section 2.1.3 and adaptive properties in Section 2.1.4. We then give an overview of concepts from developmental science in Section 2.1.5.

2.1.1 Neurobiological and Evolutionary Preliminaries

As stated in Chapter 1 an interesting property of many animal species is their similarity with respect to the fundamental layout of their information processing, i.e., their nervous system. A good introduction to this topic and the various components of the nervous systems is given in [Ghysen, 2003]. It points out the similarities between the nervous systems of various species and discusses the possibility of a common ancestor where [Jékely, 2011] discusses the development sensory-motor neurons and the benefit of such a tight coupling from an evolutionary perspective with respect to cost reduction. [Silver, 2010] gives an overview of the computational capabilities of neural circuits where [Faisal et al., 2008] gives an overview of the sources and of noise in the nervous systems and discusses its benefits for information processing.

2.1.2 Entorhinal Cortex and Hippocampus

One of the fundamental capabilities of organisms is their ability to traverse their environment in a goal directed fashion, i.e., while searching for food, evading predators or finding their way back home. The question remained on the neurobiological foundations of these capabilities, i.e., the question on how spatial relationships and the environment were represented. Though experiments with rats provided evidence for an internal representation of space that facilitates the use of spatial relations, e.g. for calculating shortcuts [Tolman, 1948] the breakthrough was the discovery of place cells in the rat's hippocampus presented in where the activity of such a place cell is related to the position of the animal in its environment. [O'Keefe and Nadel, 1978, pp. 1–2] proposed the hippocampus as a cognitive map, stating that *"there exists at least one neural system which provides the basis for an integrated model of the environment. This system underlies the notion of absolute, unitary space, which is a non-centered stationary framework through which the organism and its egocentric spaces move. We shall call the system which generates this absolute space a cognitive map and will identify it with the hippocampus."* In addition [O'Keefe and Nadel, 1978] propose that similar structures might exist in humans and are the foundation of spatial representations. Other specialized types of cells associated with the representation of space have since been discovered like head direction cells, cells whose activity is specific to a certain orientation of the animal's body [Taube et al., 1990a, Taube et al., 1990b]. Another type of cell are grid cells proposed in [Fyhn et al., 2004] and described in more detail in [Hafting et al., 2005]. While place cells allow for the representation of distinct places, grid cells are not associated to a specific location but instead are organized in a regular triangular pattern. They are supposed to be an integral system required for path integration, i.e., the mechanism that allows for the calculation of the position in space based on sensory and motor cues [Jeffery and Burgess, 2006, Mcnaughton et al., 2006]. While the existence of similar structures in the human brain has been assumed [Doeller et al., 2010] could finally show the existence of grid cells in the human brain using functional magnetic resonance imaging (fMRI) on subjects performing tasks in virtual environments. Additional types of cells, denoted 'border cells', have been discovered in the rat by [Solstad et al., 2009] where the response of these cells corresponds to the spatial proximity to a boundary in the environment. [Bjerknes et al., 2014] discusses the role these border cells play in the representation of space and especially their relation to place cells. They propose that border cells might serve as an initial mechanism

for path integration stating that *"spatial signals from border cells may be sufficient to maintain spatially localized firing in juvenile hippocampal place cells"*. This is based on the observation that grid cells develop slower and are initially not capable of providing stable signals as their responses are *"irregular and variable in size and shape"*. As most of the experiments, both with rats and with human subjects, have been conducted in two-dimensional space, data on how three-dimensional space is represented is somewhat scarce [Ulanovsky, 2011]. Findings of experiments that have been conducted with flying bats are discussed in [Yartsev and Ulanovsky, 2013] that show the responses of place cells in the bat's hippocampus with respect to the bat's position in three-dimensional space. However, many questions still remain open with respect to the relation between the topology of the environment and the topology of the neural structures which has been stated by [O'Keefe et al., 1998] as *"there is no topographical relation between the anatomical location of the cells within the hippocampus and the place fields of these cells in an environment"*. Another question is whether the role of the hippocampus is specifically associated to the representation of space or whether it is associated to a broader array of memory tasks. [Schiller et al., 2015] discusses this question and proposes a unifying view that sees the hippocampus as a morphological structure of the brain that is capable of *"organizing incoming information within the context of a multidimensional cognitive map of spatial, temporal, and associational context."*

2.1.3 Sensory and Sensorimotor Integration

An interesting property of the nervous system is the tight coupling of sensory and motor components. In [Wexler et al., 1998] experiments were conducted in which human subjects were presented a joint mental and motor rotation task where the findings showed interferences between the two modalities. Experiments conducted in [Graziano, 1999] where a fake arm was utilized to present inconsistent proprioceptive and visual cues to primates suggests that visual and proprioceptive information is integrated into a consistent internal representation. In this regard [Fadiga et al., 2000] promotes the interpretation that the *"most likely interpretation is that the neuron discharge of visuomotor neurons is neither purely visual nor purely motor: it codes a potential motor action, completely devoid of impending motor requirements or, according to an old terminology, it represents the idea of a specific action"*. The close relation between mental motor representation and actual execution of motor actions is discussed in [Kerzel et al., 2001] where experimental findings suggest that *"perception and action seem to be fed by a common, cognitively penetrable, spatial representation"*. A review of sensory integration can be found in [Ernst and Blthoff, 2004] while [Huston and Krapp, 2008] discuss the coupling between the sensory and the motor system in flies with respect to the task of gaze stabilization.

2.1.4 Adaption

A topic related to the integration is that of adaption, i.e., the question on how information processing is altered in situations where certain properties of the system change. [Jarvis et al., 2013] present the biomechanical changes observable in dogs after amputation of a thoracic limb with a focus on the changed gait patterns where [Della-Maggiore et al., 2015] gives a good overview of

the topic of sensorimotor adaption and further examples on the plasticity of the central nervous system.

2.1.5 Cognition and Psychology

While the neurobiological layout provides the physical foundation for spatial representation and sensorimotor integration, another important part of a biological system is how these elements are utilized, i.e., how the behavior of the system uses the available resources. With respect to the topic of this thesis an important aspect is the question of learning, i.e., how biological entities use their sensorimotor capabilities to learn about themselves and their environment. Influential research in the domain of developmental psychology has been conducted in [Piaget, 1953] where the main topic is the development of children especially with respect to their sensorimotor capabilities. Six behavioral stages are proposed and we give a brief overview of the first three. While the first behavioral stage is associated with simple reflex-based actions, the second and third introduce an intentional component to the child's behavior where [Piaget, 1953, p. 55] states that in *"characterizing these acquisitions it must also be noted that they imply an active element. It is not a question of associations imposed by the environment, but rather of relationships discovered and even created in the course of the child's own searchings. It is this twofold aspect of acquisition and activity which characterizes what we shall henceforth call 'circular reactions'"*. In the second stage, the 'primary circular reaction', previously unorganized reflex-based actions are coordinated and executed with the aim to recreate a previously encountered beneficial state while taking into consideration only the own body. In the third stage, the 'secondary circular reaction', these actions involve the environment, stated by [Piaget, 1953, p. 154] as: *"After reproducing the interesting results discovered by chance on his [sic] own body, the child tries sooner or later to conserve also those which he [sic] obtains when his [sic] action bears on the external environment."* While [Piaget, 1953] focused on early sensorimotor learning, i.e., the sensorimotor properties of the individual with respect to itself and distinct objects, [Siegel and White, 1975] conducted experiments regarding the influence of motor exploration in the process of learning the representation of large spatial environments where the main finding was that the accuracy *"of construction improved as a function of motor experience"*. Another interesting effect with respect to the early development of spatial and sensorimotor representation is that a notion of consistency with respect to certain properties of the physical world seems to be learned at an early age. Experiments with infants in [Baillargeon, 1994] showed that even at an early age infants seemingly detected inconsistent situations like objects defying gravity or disappearing. An overview of more recent findings can be found in [Baillargeon et al., 2010].

2.2 From Biology to Technical Systems

In this section we give an overview of certain concepts from biology that served as an inspiration for the development of models from the domain of computer science and robotics.

2.2.1 Neural Networks

One of the major bio-inspired approaches in this regard was the development of artificial neural networks (ANNs) introduced by [McCulloch and Pitts, 1943] where these first networks were composed from artificial neurons modeled by means of propositional logic. Since then many new concepts, models and applications have been developed and we refer to [Russell and Norvig, 2003, Chapter 20.5] for a brief introduction to the topic and to [Rojas, 1996] for a comprehensive overview. A fundamental property of ANNs is their capability to model the functional relation between an input and an output space by learning from a provided set of training data where, once trained, such a network provides an inherent capability of generalization. Building on this fundamental concept [Broomhead and Lowe, 1988] proposed that *"generalization is therefore synonymous with interpolation"* which *"led to the theory of multivariable interpolation in high dimensional spaces"*. They introduced a new type of ANN based on radial basis functions where a good introduction and a review of different applications for these types of networks can be found in [Wu et al., 2012]. Inspired by evidence that *"topographic maps of the exterior environment are formed in the hippocampus"* and that *"some areas of the brain could simply create and order specialized cells or cell groups in conformity with high-level features and their combinations"* a new type of ANN denoted 'self-organizing maps' (SOM) was proposed in [Kohonen, 1982]. Self-organizing maps are an unsupervised approach for dimensionality reduction that establishes a lower-dimensional map in the input space while detecting and preserving the topological relations within the data.

2.2.2 Particle Swarm Optimization

The approaches presented in Section 2.2.1 were based on neurobiological concepts, networks constructed from artificial neurons that were inspired by their biological counterparts. Other approaches exist that are inspired by the joint behavior of multiple entities. A notable bio-inspired approach for derivative-free function approximation is proposed in [Kennedy and Eberhart, 1995] denoted 'particle swarm optimization' (PSO). It originated from the simulation of swarm behavior, e.g. flocks of birds or schools of fish, and simulates a set of agents moving through the search space. The position and velocity of these agents as well as their interaction is based on a set of simple rules that allow the artificial swarm to eventually converge towards a (local) optimum. A good introduction to PSO techniques can be found in [Poli et al., 2007]. where an overview of PSO variants can be found both in [Imran et al., 2013] and [Parsopoulos, 2016].

2.2.3 Motor Babbling

While the PSO approach is based on the behavior of multiple interacting entities, other approaches are inspired by findings from developmental psychology. One of these approaches is denoted 'motor babbling' and is inspired by the work proposed in [Piaget, 1953] (see Section 2.1.5) where the basic idea is to mimic the behavior of infants exploring and learning about their sensory and motor capabilities. Thus 'motor babbling' is the concept to utilize random or reflex-based actions for controlling a virtual or physical robotic system, thus generating ran-

dom data points from which the system can infer and construct an initial representation of its sensorimotor capabilities. Examples for systems that utilize motor babbling to learn a mapping between sensory inputs and motor controls are given in [Caligiore et al., 2008], [Sturm et al., 2008], and [Saegusa et al., 2009] where the latter extends the basic concept of motor babbling using only random controls by introducing a confidence function that allows the system to actively select controls that allow it to improve its internal model. A similar approach is proposed in [Najjar and Hasegawa, 2013] where the ratio between random motor controls, i.e., invoking motor babbling behavior and task-related actions is based on an internal error measure. This approach allows the system to react to changes in the sensorimotor interaction with its environment by dynamically adapting and re-training the previously learned sensorimotor mapping. Refering back to [Piaget, 1953], both the approach proposed in [Saegusa et al., 2009] and [Najjar and Hasegawa, 2013] can be seen as an adaption of the previously mentioned primary and secondary circular reaction as both approaches add an element of intention to the process that generates the next motor control. Another interesting approach proposed in [Marjaninejad et al., 2019] combines the bio-inspired model of behavior, i.e., motor babbling with a tendon-driven robotic limb thus hardware that was inspired by biomechanical principles.

2.2.4 Active Exploration

In contrast to the concept of motor babbling, where actions are selected randomly, active exploration describes the strategy of selecting actions that are expected to have a beneficial input with respect to some utility or performance measure that is applied to the internal representation of the agent's environment. A typical example would be a robotic system, assigned with the task of establishing a map of its environment where a corresponding utility would corresponds to coverage. Instead of randomly moving through the environment, by change exploring areas that have already been visited, a better strategy would be to actively move to unknown areas. An example for active exploration with respect to the task of self-localization is proposed in [Reineking et al., 2010] where an agent traverses a hierarchically structured environment. Actions correspond to either macro-level motor controls that facilitate movement through the environment or to micro-level saccadic eye movements with respect to a local view. Actions are selected with the aim to minimize uncertainty about the current location, i.e., where the preferred action is the one that has the highest expected information [Schill, 1997]. A related approach is presented in [Nakath et al., 2014] where active exploration is applied to the task of object recognition. A related scenario is presented in [Nakath et al., 2016] where the concept of active exploration is utilized for long-term planning of the orientation of a spacecraft's sensors during interplanetary flight. In [Nakath et al., 2019] active exploration is applied to a hybrid localization and mapping scenario with respect to a spacecraft orbiting an asteroid with the aim of establishing a three-dimensional map of the asteroid's surface. In this context the agent has to choose from two concurrent behaviors: active exploration that aims on extending the map by examining previously unvisited areas and active localization that aims on improving the current estimate of the spacecraft's position by examining regions previously visited.

2.3 Fundamental Concepts

In this section we introduce concepts that are integral to the work presented in this thesis where Section 2.3.1 covers approaches for the representation of space, Section 2.3.2 introduces the bootstrapping scenario, Section 2.3.3 gives a short introduction to manifolds and Lie groups as a special kind of manifolds where Section 2.3.5 presents protein folding and molecular docking as a conceptually related domain.

2.3.1 Map Representation and Navigation

In this section we present bio-inspired approaches that aim to provide technical systems with the means of navigation. While all the approaches presented in this section are inspired by biological principles, they approach the question from two distinct angles. Approaches that can be denoted 'bottom up' aim on modeling navigation capabilities by replicating the physiological neurobiological foundations, i.e., by explictly modeling components of the entorhinal cortex and the hippocampus (e.g. place cells and grid cells as described in Section 2.1.2). Approaches that can be denoted 'top down' aim on modeling navigation capabilities by replicating the cognitive foundations, i.e., by establishing models for representation of spatial environments based on findings from developmental psychology.

2.3.1.1 Neurobiology-Based Approaches (Bottom-Up)

A bio-inspired approach to navigation based on place cells is proposed in [Arleo and Gerstner, 2001] where both visual input and motor controls are combined in a ANN composed of place cells representing the environment connected to motor cells that generate corresponding behavior. The system learns the mapping between sensory inputs and motor controls which then can be utilized for navigation. [Milford et al., 2004] proposes RatSLAM, a approach for simultaneous localization and mapping (SLAM, see Section 2.4.1) in real time that is based on a model of the rat's hippocampus. Pose cells integrate position and orientation information, thus combining the concept of place cells and head direction cells. Though it is stated that this approach *"differs significantly from other models of the rodent hippocampus"* this integration has been performed with the aim to allow the representation of concurrent pose estimates. Another bio-inspired model for representing spatial environments and execute navigational tasks is presented in [Banquet et al., 2005]. It is composed from three layers responsible for detection of landmarks, landmark-based self-localization, path-integration and navigation, the latter by modeling spatiotemporal relations, i.e., transitions between artificial place fields.

While the approaches presented so far aim to adapt certain navigational behaviors by mimicking neurobiological structures, other approaches try to explain biological properties by establishing computational models. [Guanella et al., 2007] proposes an ANN model of grid cells where individual cells are distributed on a twisted torus where this spatial configuration of the neural network allows the representation of the regular triangular patterns associated to the spatial configuration of grid cells [Hafting et al., 2005]. A similar approach is presented in [Mhatre and Gorchetchnikov, 2012]. It proposes the GRIDsmap model that consists of a number of hierarchically organized SOMs and a special type of cell denoted 'stripe cell' that encodes distance

traveled in a given direction as a one-dimensioal signal. Based on simulations with trajectories of a virtual rat they propose that the hexagonal spatial layout of grid cells is a result of overlapping firing patterns of these stripe cells.

2.3.1.2 Cognition-Based Approaches (Top-Down)

Considering the properties of the representation of spatial knowledge with respect to findings from developmental psychology [Kuipers, 1982] discusses the applicability of the metaphor of the *"Map in the Head"* and proposes a different view, namely that of an atlas *"with each sheet defining a separate frame of reference within which any two vectors could be compared. Relative position vectors taken from different sheets could be compared only with effort, if at all, even by an adult with the normal capacity to store and retrieve vectors on the same sheet. Points on the different sheets might still be topologically related, connected by pieces of string, as it were, so that the storage and retrieval functions for connection and order could ignore the separation of the map into sheets."*. This concept is built upon in a thought experiment given in [Kuipers, 1983] where resource limitation is the driving factor for establishing an efficient representation that combines both topological and metric maps to construct a joint framework for spatial representation. A continuation of this idea presented in [Kuipers and Levitt, 1988] proposes an approach for navigation in large-scale spaces based on a hierarchical model. This model, denoted the spatial semantic hierarchy (SHH) is described in [Kuipers, 2000] and [Remolina and Kuipers, 1998] as an *"ontological hierarchy of representations for knowledge of large-scale space"*. It is comprised of five hierarchically organized levels (sensorimotor, control, causal, topological, metric) where each level corresponds to a different concept of space. The lower two levels (sensorimotor, control) are associated with the movement of the agent in the environment. While the sensorimotor level accounts for the invocation of sensors and actuators, the control level defines strategies (control laws) that allow the agent to move to a distinctive state in its local neighborhood (hill-climbing) or traverse from one distinctive state to another (trajectory-following). The causal level serves as an abstraction layer between the continuous representation of sensorimotor interaction given by the sensorimotor and the control level and a topological representation of the environment by associating the transition between distinct states with discrete actions. The topological level established a graph-based representation of the environment where the metric level allows establishing a metric map within a global frame of reference. Thus the SHH creates hybrid spatial representation where quantitative *"spatial information is represented at each level of the hierarchy, from local analog maps at the control level, to action magnitudes at the causal level, to local headings and distances at the topological level. This is enough to represent a 'patchwork metrical map' of local frames of reference linked by a topological network structure."* Thus the SSH can be seen as a distinct implementation of the atlas proposed earlier in [Kuipers, 1982]. In contrast to this complex hierarchical model, [Mallot et al., 1995] proposes a view graph as a foundation for navigation in simple environments. A review of various biology inspired approaches to robot navigation is given in [Franz and Mallot, 2000]. A classification schema is developed with respect to two concepts: navigation and wayfinding. With respect to the former it proposes a set of hierarchically structured behaviors corresponding to searching, direction-following, aiming and guidance, where each level corresponds to an increasingly complex task. Wayfinding behaviors are defined as

recognition-triggered responses, topological navigation and survey navigation where the latter requires the entity to make geometric interpretation to infer about shortcuts. A differently structured hierarchical representation of spatial environments is proposed in [Zetzsche et al., 2008] that is based on sensorimotor features. A sensorimotor feature is a 3-tuple that describes the relation between two sensory inputs perceived at different states with respect to a motor action that encodes the corresponding state transition. In this regard it is closely related to the concept of views and actions described in [Kuipers and Levitt, 1988] where a view *"represents the travelers sensory input at a given instant"* and an action is defined to correspond to *"a change of state, with the current view defined immediately before and after an action but not changing continuously during it"*.

An interesting question with respect to the representation of spatial knowledge is the relation between metric and topological information and to which extent a global metric representation is constructed and utilized when performing navigational tasks. In [Kluss et al., 2015] subjects were assigned the task to explore virtual non-Euclidean corridor-shaped environments, i.e., environments that violated certain properties of Euclidean space with respect to distances and/or angles. After an exploration phase the subjects were assigned the task of reproducing the traveled paths in a neutral virtual setting. In contrast to the non-Euclidean properties of the environment *"analyses of recalled route data do not indicate angular or configurational distortions"*. These results are in line with the findings proposed in [Warren et al., 2017] where subjects had to explore and navigate through a virtual maze that contained 'wormholes', points in space that connected spatially unrelated locations of the maze and additionally introduced a rotational component. Confronted with several navigational tasks, subjects were able to traverse the maze despite its non-Euclidean nature. Interestingly *"participants were completely unaware of these geometric inconsistencies, reflecting a surprising insensitivity to Euclidean structure"*.

2.3.2 Bootstrapping

In this section we discuss the concept of bootstrapping, i.e., the scenario in which an agent is provided with little to no a priori information about itself or its environment and is confronted with the task of establishing a suitable representation of itself and its environment that can be utilized for deliberative actions where according to [Censi and Murray, 2015] it *"can be seen as an extreme form of system identification/calibration"*. A good introduction to this scenario is given in [Pierce and Kuipers, 1997] where it is described as *"a robot with an uninterpreted, almost-everywhere approximately linear sensorimotor apparatus in a continuous, static environment"* where the objective is to learn the *"descriptions of the structure of the robots sensorimotor apparatus and environment and an abstract interface to the robot suitable for prediction and navigation"*. Interestingly a distinction is made between the agent and the robot, the latter as being described as *"a machine (physical or simulated) that the learning agent must learn how to use"*. Closely related to the hierarchical concept of the SHH [Kuipers, 2000] it proposes an approach for successively constructing an increasingly complex hierarchical model of the agent's sensorimotor capabilities. The approach is proposed for a simulated robot situated in a bounded two-dimensional environment. The robot is equipped with an omnidirectional distance sensor proving a 24 dimensional feature vector, a digital compass and a sensor that returns the charge of the robot's battery where all sensors returned values between 0.0 and 1.0 and none of the sensors

were subject to noise. The actuator of the robot allowed for individual control the left and the right wheels, thus enabling it to turn on the spot, move in circular arcs or in a straight line. In the bootstrapping scenario motor controls are created randomly, creating a random sequence of motor controls and associated sensor values. In a first step sensors are grouped together based on similarities in their output, assuming that the measurements of sensors in close proximity should be similar where based on these groups their spatial layout is computed. In the next step systematic motor controls are executed where the effect of the motor controls on the sensor values is represented in terms of vector fields where principal component analysis was used to find a set of 'basis' vector fields. Associating the basis vector fields to the motor controls allows for representing basis motion capabilities with respect to the change in sensory perception. Based on this abstract actuator model the agent then associates motor controls to sensory inputs with the aim to find causal relations between motor controls and sensory features that allow the agent model control laws, i.e., homing- and path-following behaviors. Extensions to that approach are presented in [Kuipers and Beeson, 2002], [Provost et al., 2006] and [Kuipers et al., 2006], proposing mechanisms for place recognition and utilizing self-organizing maps for learning of distinctive states. A related approach building upon the work in [Pierce and Kuipers, 1997] is proposed in [Olsson et al., 2005] and [Olsson et al., 2006] allows an agent to detect fundamental relations between visual inputs and motor controls and to utilize these rules for performing motion tracking.

A slightly more philosophical view on the bootstrapping scenario with respect to visual perception is proposed in [ORegan and No, 2001] which proposes that *"seeing is a way of acting"* and introduce sensorimotor contingencies (SMC) where the *"experience of seeing occurs when the organism masters what we call the governing laws of sensorimotor contingency"*. These sensorimotor contingencies are defined as the *"structure of the rules governing the sensory changes produced by various motor actions"*. Thus SMC are in a way comparable to the basis vector fields [Pierce and Kuipers, 1997] identified by relating motor controls to the changes in sensory inputs. [Buhrmann et al., 2013] builds upon the basic concepts proposed in [ORegan and No, 2001] and defines four distinct kinds of sensorimotor structures denoted sensorimotor environment, sensorimotor habitat, sensorimotor coordinations and sensorimotor strategies. Here sensorimotor environment refers to the change in sensor inputs, induced by the execution of motor controls. This kind of sensorimotor structure is *"independent of the agent's internal state"* and is modeled as a functional relation between the agent and its environment, thus it *"constitutes the set of all possible sensory dependencies on motor states* (s, m) *for a particular type of agent and environment"*. The sensorimotor habitat describes the *"set of all sensorimotor trajectories"* that can be executed by an agent situated in an sensorimotor environment, given a closed-loop relationship between both. Sensorimotor coordinations are defined as *"trajectories within the SM habitat that occur reliably and contribute functionally to a goal"* where a sensorimotor strategy is defined as *"an organization of SM coordination patterns that is regularly used by the agent because it has been evaluated as preferable (along some relevant normative framework) for achieving a particular goal"*. Thus sensorimotor coordinations can be seen as atomic components that can be utilized for creating more complex plans, i.e., sensorimotor strategies.

The concepts proposed by [ORegan and No, 2001] have been utilized in [Philipona et al., 2003] where this work has been continued and extended in [Philipona et al., 2004]. These

approaches model the sensorimotor interaction of an agent with unknown sensors and actuators with its environment using Lie groups, a special kind of manifold (see Section 2.3.3) allowing the agent to detect the dimensionality of the environment it is situated in. A related approach is proposed in [Laflaquire et al., 2015] where, other than in the previous approaches, it focuses on detecting the topological structure of the environment from the sensorimotor interaction.

2.3.3 Manifolds

Manifolds are topological spaces that locally resemble Euclidean space, i.e. an n-dimensional manifold locally looks like an n-dimensional Euclidean space where globally this resemblance does not have to hold. Manifolds can be embedded in higher dimensional spaces [Whitney, 1944] where an example of such an embedding would be the earth as a 2-dimensional manifold being embedded in 3-dimensional Euclidean space. These properties are especially interesting in scenarios where valid configurations of a system lie on a lower-dimensional structure embedded in a higher-dimensional feature space. The task of manifold learning thus denotes the process of finding such a lower-dimensional structure based on a set of data points sampled from the manifold but lying in the higher-dimensional space where a good overview of manifold learning algorithms is given in [Huo and Smith, 2008] and [Lin and Zha, 2008]. An interesting approach is presented in [Pitelis et al., 2013] where an unknown manifold is learned as an atlas, i.e., a set of overlapping charts which is applied to the problem of reconstructing human motion from two-dimensional camera images.

Manifolds can be utilized within the context of motion generation where [Berenson et al., 2009] proposes a manifold based approach to robot motion planning that utilizes a tree-based planning algorithm to find bridges between manifolds that represent constraints in the robot's configuration space. A related approach is presented in [Havoutis and Ramamoorthy, 2010b] that proposes skill manifolds, manifolds that encode behavior patterns that can be used for robot motion generation where individual motion patterns correspond to geodesics (locally shortest paths, see Section 3.4) on this manifold. An extension to that approach that incorporates constraints is presented in [Havoutis and Ramamoorthy, 2010a]. A similar approach is presented in [Jaillet and Porta, 2013]. Here a manifold is not utilized as a tool for dimensionality reduction. Instead an atlas is created as a set of overlapping local Euclidean charts and then utilized to perform path planning.

2.3.4 Lie Groups

Lie groups are a special kind of manifold that exhibit both manifold and group properties where a more thorough introduction will be given in Section 3.5. Lie groups are an elegant tool in scenarios where the manifold is traversed as due to qualities provided by the group property, it is not possible to 'leave' the manifold as operations are guaranteed to 'move on it'. This makes these structures well-suited for motion planning and representation of robot kinematics. A good introduction on Lie groups can be found in [Stillwell, 2008] where an introduction to the general subject of robot motion planning can be found in [Choset et al., 2005]. An example for using Lie groups in the domain of computer vision and animation of vehicles is given in [Kobilarov et al., 2009] that proposes Lie group based controller and shows the applicability of Lie groups for

representing transformations that facilitate the animation of natural motions. Examples from the domain of robot controls are given in [Nordkvist and Sanyal, 2010] that proposes an integrator for an underwater vehicle and [Nakath et al., 2017] that proposes a feedback controller for the attitude control of a simulated spacecraft [Probst et al., 2015]. Though systems are often considered as being drift-free, i.e., don't account for external forces acting upon the system, real-world applications are often modeled with additional drift-forces. An example is given in [Sarti et al., 1993] that proposes a Lie group based approach for compensating these drift forces and [Walsh et al., 1994] that proposes an approach to optimal trajectory planning of an airplane that is subject to drift. Apart from the domain of robot motion control Lie groups can be utilized in action recognition where [Huang et al., 2017] demonstrates the combination of ANN with Lie groups by proposing a deep learning architecture that directly encodes the Lie group structure into the network layers and is utilized for skeleton-based action recognition.

2.3.5 Molecular Docking and Protein Folding

The problem statement of molecular docking is to find a relative spatial configuration that allows a given molecule to 'dock' to another where a valid docking configuration is characterized as minimizing an energy function that is based on intra- and intermolecular properties. The difficulty in this task is that not only the relative spatial configuration between these two molecules can be altered but that the internal spatial configuration of molecules is flexible as well. From a geometric point of view, a molecule can be seen as a flexible structure where rigid segments are connected to each other by rotatable joints. Finding a suitable docking configuration thus corresponds to the task of finding a configuration for these joints that minimizes a given energy function. An introduction to that topic is given in [Teodoro et al., 2001] while [Mirzaei et al., 2015] proposes a manifold based approach where the spatial configuration of molecules is represented in terms of rotation and translation. Representing these configurations with respect to the tangent space of the manifold allows for utilizing standard optimization algorithms (LBFGS) for minimizing an energy-based cost function. A related topic is that of protein folding where, instead of searching only for a valid end configuration the intermediate states are considered as well. [Amato and Song, 2002] proposes an approach that utilizes concepts from robotics motion planning. Proteins are modeled as a multi-joint tree-like robots where atomic bonds and atoms correspond to joints and links respectively. Finding a valid folding sequence thus corresponds to finding a path through the 'robot's' configuration space where configurations are evaluated with respect to a function that calculates its potential energy with the aim on minimizing it.

2.4 Related Concepts

In this section we briefly present concepts and approaches that are not directly utilized within this thesis but are closely related to the topics of navigation, bootstrapping or underline the requirement and current state on autonomous systems. Section 2.4.1 gives a brief overview of technical approaches related to the task of navigation where Section 2.4.2 gives an overview of autonomous systems and robotics.

2.4.1 Navigation

As previously stated in Chapter 1, according to [Leonard and Durrant-Whyte, 1991] navigation deals with three question, namely *"where am I?"*, *"where am I going?"* and *"how should I get there?"* where these three questions can be associated to the tasks of (a) localization and mapping, (b) deliberation and reasoning and (c) path planning and control respectively. In this section we mainly discuss approaches associated to localization and mapping, as both deliberation and path planning are not the main focus of this thesis. Mapping refers to the task of establishing a representation of the envionment that can serve as a frame of reference where localization denotes the closely related task of determining the relation between an agent and said representation, e.g. identifying the agent's position and orientation. Both tasks can be approached individually, i.e., establishing a map based on sensor measurements whose locations are known or determining the location of an agent relative to a known map of the environment. However, in many cases both localization and mapping have to be performed simultaneously, a scenario accordingly denoted simultaneous localization and mapping (SLAM).

With respect to map representations, multiple types exist where according to [Thrun and Bücken, 1996] two distinct types of map representation for indoor environments can be distinguished. Topological approaches *"represent robot environments by graphs. Nodes in such graphs correspond to distinct situations, places, or landmarks (such as doorways). They are connected by arcs if there exists a direct path between them"* where grid-based (or metric) approaches *"represent environments by evenly-spaced grids. Each grid cell may, for example, indicate the presence of an obstacle in the corresponding region of the environment"*. [Thrun et al., 2005, p. 152] proposes a slightly more differentiated definition for metric maps. Here, location-based maps are proposed as *"volumetric, in that they offer a label to any location in the world"* and that they *"contain information not only about objects in the environment, but also about the objects"* where feature-based maps *"only specify the shape of the environment at the specific locations, namely the of the objects contained in the map"*. Examples for approaches that utilize purely topological maps without metric information are given in [Franz et al., 1998] and [Fraundorfer et al., 2007] where both approaches establish topological maps of the environment based on sequences of images and utilize these graph-based representations for navigation. Grid maps, i.e., occupancy grid maps, are commonly used for robot navigation. However, different approaches exist with respect to the representation of uncertainty. [Grimme et al., 2017] proposes the usage of belief functions [Shafer, 1976], [Lang et al., 2007] utilize Gaussian processes [Rasmussen and Williams, 2005], and [Ramos and Ott, 2016, Senanayake and Ramos, 2017] establish a new representation denoted Hilbert maps for representing uncertainty.

With respect to SLAM a good introduction to the topic can be found in [Thrun et al., 2005, Chapter 10] where [Frese et al., 2010] provides an overview of different SLAM approaches. Notable examples utilize occupancy grids [Thrun, 2003], use topological maps [Choset and Nagatani, 2001], model the environment using splines [Pedraza et al., 2009], use graph-based representations [Grisetti et al., 2010, Bichucher et al., 2015], or a combination of topological and metric maps [H. Jacky Chang et al., 2007].

2.4.2 Autonomous Systems and Robotics

One motivation for developing autonomous systems, i.e., equipping technical systems with the means to autonomously reason about their actions and execute them, is making them more independent from human control. With respect to the context of space exploration, [Starek et al., 2016] gives an overview of the current state-of-the-art of autonomous systems and discusses challenges and requirements for future space missions. It identifies autonomous *"maneuvering, especially in proximity of artificial objects (e.g. satellites, debris, etc.) or Solar system bodies (e.g. asteroids, comets, irregular satellites, etc.)"* as a *"key enabler for most future NASA missions"*. One major aspect in this regard are the long communication delays. Landing a spacecraft on the surface of Mars is stated as *"one of the most tightly-constrained control sequences in spaceflight achievable by current technology"* where control *"of the lander spacecraft by remote operation is simply impossible due to a nearly 26 minute two-way signal communication time, a duration far greater than the seven minutes required for the entire atmospheric Entry, Descent, and Landing (EDL) process"*. Another aspect is the *"desire to increase mission frequency, robustness, and reliability"* where utilizing autonomous systems aims on reducing the increasing costs caused by *"scheduling conflicts and increases in maintenance and labor"* as well as the inevitable increasing chance for human error.

Historically Deep Space One was the first spacecraft ever to be controlled by an autonomous agent during interplanetary travel. The Remote Agent proposed in [Muscettola et al., 1998] was a planning and scheduling component that has been developed within the context of the Remote Agent Experiment (RAX) [Jonsson et al., 2000] where the aim of RAX was to demonstrate *"closed-loop planning and execution"* as well as *"model-based state inference and failure recovery"*. Plans created by the RAX Planner/Scheduler (RAX-PS) were represented as constraint networks that were created from dynamic models of the spacecraft and allowed the system to build *"concurrent plans with over a hundred tasks within the performance requirements of operational, mission-critical software"*. This first demonstration of an autonomous agent, not only capable of performing in a proof-of-concept scenario in a controlled laboratory environment, but within the context of a real space mission, paved the way for autonomous system components to be utilized in space exploration. A comprehensive overview of the degree of autonomy utilized in Mars exploration rovers is given in [Bajracharya et al., 2008], which discusses techniques that were incorporated in Sojourner, Spirit and Opportunity where [Welch et al., 2013] gives an overview of the Curiosity rover. Other approaches for the autonomous navigation of rovers on mars are given in [Ingrand et al., 2007] and [Estlin et al., 2007] where both have been tested on real hardware in comparable scenarios on earth.

Bridging the gap between biological and technical systems, approaches exist that aim on replicating the adaptability of biological systems to external influences, increasing the robustness of technical systems. Within the context of mars exploration, [Chahl et al., 2003] proposes a bio-inspired horizon sensor for flight stabilization based on the simple eyes of dragonflies that has been tested both in an open and a closed loop scenario. [Plice et al., 2003] proposes biological inspired approaches for autonomous aerial vehicles that are suitable for mars exploration. Two different concepts are proposed and evaluated with respect to theoretical considerations as well as prototypical tests. Another example for a robotic system that aims on mimicking the adaptability of biological entities is given in [Bongard et al., 2006] where a bio-inspired, adap-

tive, four-legged robot is proposed. It infers about its own morphology and utilizes this knowledge to create gait patterns. If confronted with an unanticipated change to its configuration (partly removing one leg) it is able to re-configure its internal model and adapt by synthesizing an alternative forward motion. A similar approach based on a hexapod, i.e., a robot with six legs is proposed in [Cully et al., 2015] that, after changes to its configuration occurred, adapts by iteratively selecting and re-evaluating behaviors from a pre-calculated repertoire until a suitable compensating pattern has been found.

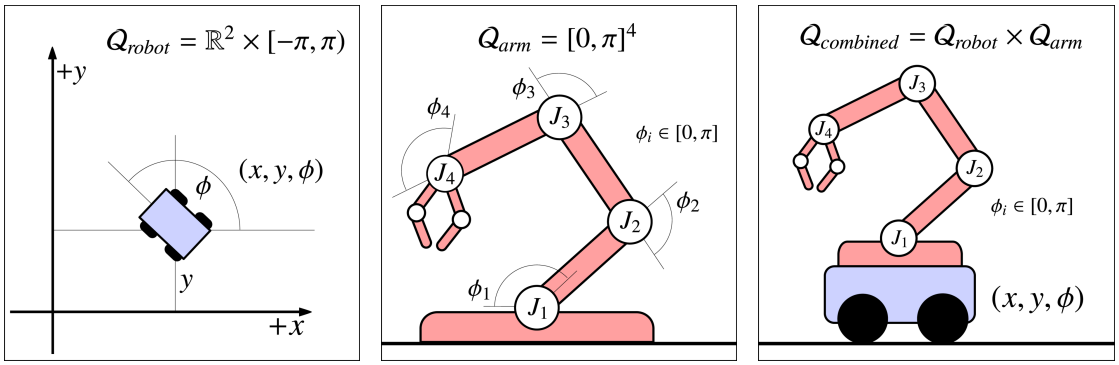
3
Mathematical Foundations

This chapter provides the mathematical foundations and the terminology essential for the definition of the sensorimotor system and agent in Chapter 4, the construction of the sensorimotor map in Chapter 5, and the development of the bootstrapping algorithm in Chapter 6, respectively. We use this chapter to provide the mathematical foundation, as well as to establish a consistent notation that we use in the remainder of this thesis. As each topic covered in this chapter is a complex subject on its own, we only give a brief introduction and otherwise refer to further reading material. The structure will be as follows: Section 3.2 introduces the concept of configuration spaces, as a way to define the state of a mobile agent in a physical world. Section 3.3 introduces manifolds as mathematical objects and motivates their usage for representing configuration spaces as well as actions, i.e., changes between configurations, while Section 3.4 introduces the concept of tangent spaces and vector fields. Section 3.5 introduces Lie groups as special kinds of manifolds and presents their unique properties, while Section 3.6 discusses the connection between Lie groups and their respective tangent spaces, denoted as Lie algebras, and motivates the utility of Lie groups for configuration space representation and action synthesis. Section 3.7 bridges the gap from manifold-based state representation to the mathematical representation of motion on manifolds, briefly introducing concepts from control theory and the terminology of holonomic and nonholonomic systems. The chapter concludes with Section 3.8, by giving a summary of the concepts and an outlook towards the next chapter.

3.1 Synopsis

In this chapter, we build the mathematical foundations for the remainder of the thesis in order to provide a foundation that allows us to formally relate a special kind of manifolds, denoted as Lie groups, to the sensory and motor capabilities of a rational agent. We start by defining the configuration space, the set of all possible configurations a mobile robot can have. Afterwards

we introduce the concept of occupied and free portions of such a space, allowing to apply this concept to scenarios where the free configuration space is only a subset of the actual space. We relate the configuration space to the concept of manifolds, objects that locally resemble Euclidean space but globally might exhibit a different structure. The aim is to formally express the effect, that a motor control has on a given configuration, i.e., model state transitions in a generic way. By expressing configuration spaces as manifolds we can relate changes of configurations to movement along the manifold. We then introduce the concept of tangent spaces as the set of all tangent vectors at a given configuration, where such a tangent vector corresponds to a certain direction and a velocity of a curve passing through that point. The definition of a tangent bundle, which is the disjoint union of all tangent spaces, allows us to introduce the concept of vector fields on manifolds. Following the flow of a vector field corresponds to changing the location on the manifold. As our configuration space has a manifold structure, we can relate vector fields to configuration changes. This allows us to express the effect of motor controls with respect to elements of the tangent space, i.e., with respect to certain vector fields. We then introduce matrix Lie groups as special kinds of manifolds that are endowed with a group structure and Lie algebras that denote their tangent space. The relation between Lie groups and their corresponding algebra is special as the tangent space at every group element is isomorphic to the tangent space at the identity element of the group. As tangent vectors correspond to vector fields on the manifold, the entirety of the Lie group structure can be described by elements from the Lie algebra. In addition, there exists a mapping from the Lie algebra to the Lie group and back, which allows to associate elements from the former to elements from the latter. These properties make Lie groups a viable foundation for representing configuration spaces, as the aforementioned properties allow us to express configuration changes, i.e., motor controls as vectors from the Lie algebra. The effect of executing such a control can be calculated by transforming the Lie algebra element to an element of the Lie group and then invoke the group operation, which yields a new configuration. Having established Lie groups as a configuration space representation, we have a closer look on which implications varying degrees of freedom have on the capabilities of a mobile entity when it comes to changing its configuration. This leads to the introduction of the concepts of holonomic and nonholonomic systems, where the former denotes a system where the degree of freedom, i.e., the number of controllable parameters matches the dimension of the configuration space, while the latter describes the case where the number of controllable parameters is lower than the dimension of the configuration space. As many technical systems are nonholonomic, we conclude the chapter by depicting the relation between the controllability of a system and the Lie group representation.



(a) Configuration space Q_{robot} of a robot living in a two dimensional Euclidean plane.

(b) Configuration space Q_{arm} of a robot arm with four revolute joints. Figure based on [Choset et al., 2005, p. 41, Figure 3.1].

(c) Combined configuration space $Q_{combined}$ as a product of Q_{robot} and Q_{arm}.

Figure 3.1: Examples for robotic systems and their configuration spaces.

3.2 Configuration Space

The notation and examples from this section are based on [Choset et al., 2005, Chapters 3.1 to 3.3].

When it comes to developing algorithms for robot navigation and control, a fundamental requirement is to be able to unambiguously and formally describe the configuration such an entity currently is in. Within the context of this thesis, we define such a configuration q as an n-dimensional vector of real numbers that sufficiently specifies the position and orientation of every part of the entity. Here, the set of all configurations is represented by the n-dimensional configuration space Q^n, where, due to it being composed of certain n-dimensional vectors from $\mathbb{R}$, it holds that

$$Q^n \subseteq \mathbb{R}^n. \tag{3.1}$$

The dimension of the configuration space $\dim(Q^n) = n$, i.e., the number of entries in each individual configuration vector q denotes the degree of freedom (DoF), which corresponds to the number of individual and independent parameters that describe the entity's configuration. Thus, the properties that individual parameters of the configuration vectors correspond to are entity-specific.

As a first example, we consider the mobile robotic system depicted in Fig. 3.1a, which resembles the one we previously described in the motivational example in Section 1.3. The configuration of this system is given by the vector $q = (x, y, \phi)$. The first two parameters x and y are the coordinates of an arbitrary fixed point of the entity in the Euclidean plane $\mathbb{R}^2$ and unambiguously define the entity's position. The third parameter ϕ defines the orientation of the entity as an angle in the range of $[-\pi, \pi)$. The configuration space of this entity is the set of all possible positions and orientations and is thus given by the Cartesian product of both individual

sets as

$$\mathcal{Q}_{robot} = \mathbb{R}^2 \times [-\pi, \pi), \tag{3.2}$$

where the dimension of the configuration space and therefore the degree of freedom of this entity is

$$\dim(\mathcal{Q}_{robot}) = \dim(\mathbb{R}^2) + \dim([-\pi, \pi)) = 3. \tag{3.3}$$

As a second example, we consider the stationary robotic arm depicted in Fig. 3.1b that has four revolute joints[1] denoted as J_1, J_2, J_3, and J_4, each capable of being set to an angle in the range of $[0, \pi]$. Therefore, a configuration for this system can be given by the vector $q = (\phi_1, \phi_2, \phi_3, \phi_4)$ where $\phi_1, \phi_2, \phi_3, \phi_4 \in [0, \pi]$. The configuration space comprises of the Cartesian product of the individual ranges of these joints, thus

$$\begin{aligned} \mathcal{Q}_{arm} &= [0, \pi] \times [0, \pi] \times [0, \pi] \times [0, \pi] && (3.4) \\ &= [0, \pi]^4, && (3.5) \end{aligned}$$

where the dimension of the configuration space, i.e., the degree of freedom is

$$\dim(\mathcal{Q}_{arm}) = 4. \tag{3.6}$$

Arbitrary combinations of both types of configuration spaces are possible, resulting in more complex mobile systems. Attaching the robotic arm to the mobile robot would lead to the combined system depicted in Fig. 3.1c with the configuration space

$$\begin{aligned} \mathcal{Q}_{combined} &= \mathcal{Q}_{arm} \times \mathcal{Q}_{robot} && (3.7) \\ &= \mathbb{R}^2 \times [-\pi, \pi) \times [0, \pi]^4, && (3.8) \end{aligned}$$

where the degree of freedom corresponds to

$$\dim(\mathcal{Q}_{combined}) = \dim(\mathcal{Q}_{arm}) + \dim(\mathcal{Q}_{robot}) = 7. \tag{3.9}$$

Though the configuration space $\mathcal{Q}^n$ comprises all configurations that are mathematically possible, in many scenarios the movement of a robotic system might be obstructed, either by limitations of its motor capabilities or by obstacles in its environment.

To be able to represent these cases, we define the entity-specific occupancy function

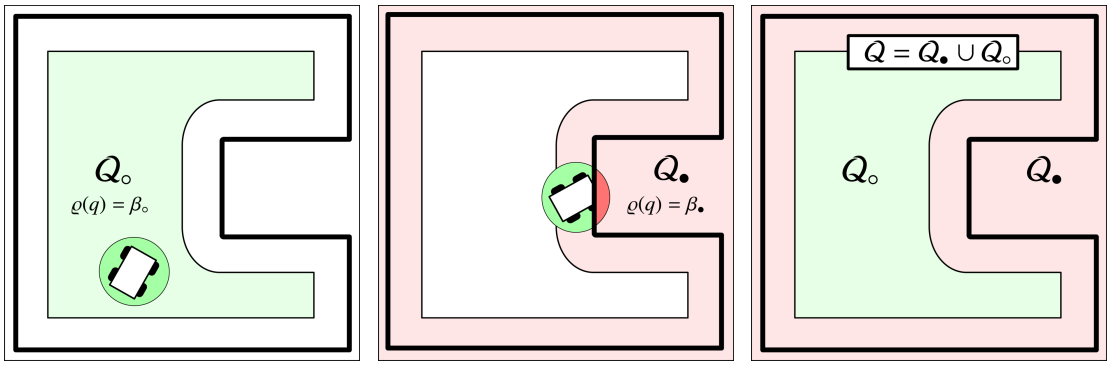
$$\varrho : \mathcal{Q}^n \to \mathcal{B} \tag{3.10}$$

that assigns to every $q \in \mathcal{Q}^n$ an element from the boundary set $\mathcal{B}$ with

$$\mathcal{B} = \{\beta_\circ, \beta_\bullet\}. \tag{3.11}$$

Here $\beta_\circ$ indicates valid, unobstructed or free configurations and $\beta_\bullet$ invalid, obstructed or occupied configurations. The occupancy function ϱ has to be entity-specific as, given the same

[1] Joint that allows rotation about a single axis, see [Choset et al., 2005, p. 49].



(a) Empty and accessible configuration space, namely the set of configurations the occupancy function ϱ returns $\beta_\circ$.

(b) Occupied and inaccessible configuration space, namely the set of configurations the occupancy function ϱ returns $\beta_\bullet$.

(c) Union of $Q_\circ$ and $Q_\bullet$.

Figure 3.2: Relation between the entity-specific occupancy function and the configuration space. Figure based on [Choset et al., 2005, p. 44, Figure 3.4 and 3.5].

environment, the size and shape of an entity has an immediate impact on the set of valid configurations. An example is a small room cleaning robot that can[2] reach places a larger vacuum cleaner might not have access to. Thus, the configuration space is composed of the two disjoint sets

$$\begin{aligned} Q^n_\circ &= \{\, q \in Q^n \,|\, \varrho(q) = \beta_\circ \,\} \quad \text{and} \\ Q^n_\bullet &= \{\, q \in Q^n \,|\, \varrho(q) = \beta_\bullet \,\}, \end{aligned} \tag{3.12}$$

denoting the empty ($Q^n_\circ$) and occupied ($Q^n_\bullet$) portions of the configuration space, respectively. For a given ϱ it holds that

$$\begin{aligned} Q^n_\circ \cap Q^n_\bullet &= \emptyset \quad \text{and} \\ Q^n_\circ \cup Q^n_\bullet &= Q^n, \end{aligned} \tag{3.13}$$

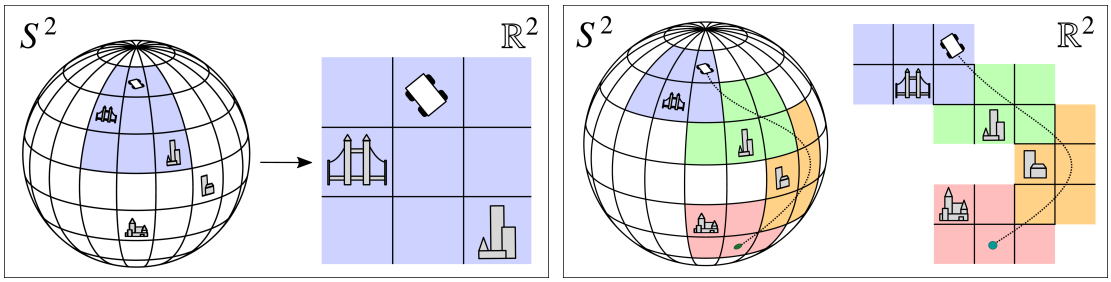
i.e., the configuration space in its entirety is composed of both empty and occupied parts. An example, based on [Choset et al., 2005, Chapter 3.2.1], is given in Fig. 3.2, while for a more in-depth discussion on configuration spaces we refer to [Choset et al., 2005, Chapter 3].

3.3 Manifolds

The notation and examples in this section are based on [Choset et al., 2005, Chapters 3.2, 3.4 and 3.5], [Bloch et al., 2007, Chapter 2.2], and [Lee, 2003, Appendix A]

So far the configuration space Q^n provides a formal foundation that allows us to represent distinct individual configurations of an entity as n-dimensional real vectors. However, we have yet

[2] And likely should!



(a) Representing a local neighborhood of S^2 as a map, a local patch that resembles $\mathbb{R}^2$.

(b) Moving across the manifold S^2 by successively traversing from one chart to another.

Figure 3.3: Visualization of S^2 as an example for a manifold. The manifold S^2 is locally homeomorphic to $\mathbb{R}^2$. This allows us to represent local neighborhoods from S^2 as two-dimensional maps.

to define how to represent a transition from a given configuration $q_i \in \mathcal{Q}^n$ to another configuration $q_j \in \mathcal{Q}^n$, which is mandatory for being able to model entity movement and ultimately the relation to motor controls. More precisely, we want the configuration space to be endowed with additional structure, so that (a) given two configurations $q_i, q_j \in \mathcal{Q}^n$ we are able to find a smooth transition that allows us to move from q_i to q_j without leaving $\mathcal{Q}^n$, and that (b) such a transition exists for all pairs $q_i, q_j \in \mathcal{Q}^n$. Mathematical spaces that exhibit these properties (among others) are called smooth - and with respect to the latter requirement, connected - manifolds. On a fundamental level manifolds are topological spaces that are locally homeomorphic to an Euclidean space $\mathbb{R}^n$, i.e., they locally look like an n-dimensional Euclidean space, while globally this resemblance usually does not hold. A visual example for a manifold is the earth which locally resembles $\mathbb{R}^2$ as its surface appears planar. A sufficiently small area or neighborhood can therefore be represented by a local two-dimensional chart in $\mathbb{R}^2$, which corresponds to the ability to draw a two-dimensional map of a sufficiently small area, e.g. a city as depicted in Fig. 3.3a. However, representing the entire surface of the earth within a single chart in $\mathbb{R}^2$ is not possible without performing a cut at some point. Nevertheless, it is possible to represent the transition from one point on the earth to another by simply traversing across an arbitrary number of individual overlapping charts as depicted in Fig. 3.3b.

Formally, an n-dimensional manifold M [Bloch et al., 2007, pp. 62–63] can be defined by a collection of coordinate charts (U_α, ϕ_α) consisting of a neighborhood $U_\alpha \subset M$ and a homeomorphism ϕ_α, mapping U_α into an open subset of $\mathbb{R}^n$, with

$$\phi_\alpha : U_\alpha \to \mathbb{R}^n, \tag{3.14}$$

where the set of all charts is called an atlas and ϕ_α is called a coordinate function. For two overlapping charts (U_α, ϕ_α) and (U_β, ϕ_β), there exists a function that allows to perform coordinate transformations, i.e., transform points from one coordinate chart $\phi_\alpha(U_\alpha \cap U_\beta)$ to another $\phi_\beta(U_\alpha \cap U_\beta)$. According to Eq. (3.14), such a transition map is given by

$$\phi_\beta \circ \phi_\alpha^{-1} : \phi_\alpha(U_\alpha \cap U_\beta) \to \phi_\beta(U_\alpha \cap U_\beta) \quad \text{given that} \quad U_\alpha \cap U_\beta \neq \emptyset. \tag{3.15}$$

An example for such a coordinate transformation with two overlapping neighborhoods U_α and U_β is depicted in Fig. 3.4. If the transition maps for a given manifold are smooth, the manifold

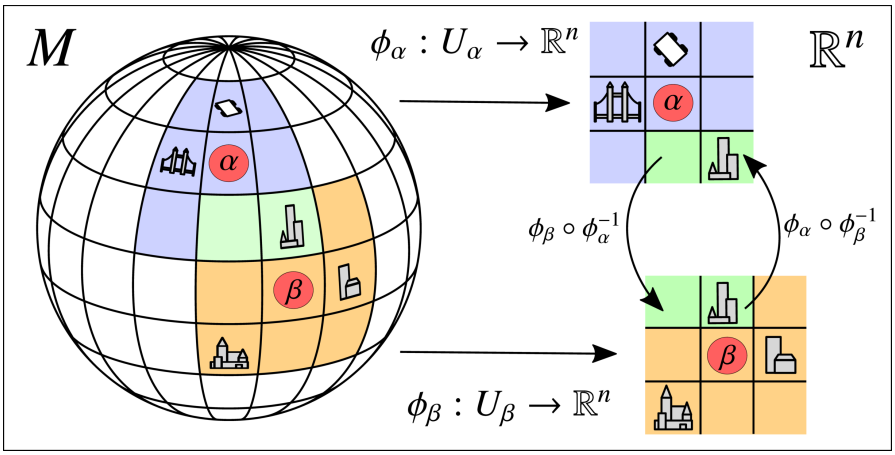


Figure 3.4: Coordinate transformation between two overlapping neighborhoods U_α and U_β. Figure based on [Bloch et al., 2007, p. 63, Figure 2.2.1] and [Choset et al., 2005, p. 57, Figure 3.13].

is considered to be smooth itself. Within the context of this thesis, we consider the configuration space Q^n to have a corresponding smooth structure. The manifold itself can be entirely represented by its atlas: overlapping coordinate charts define local neighborhoods on M, and the respective coordinate transformations provide a formal way to patch these neighborhoods together [Bloch et al., 2007, p. 63]. Thus it holds that

$$\bigcup_\alpha U_\alpha = M. \tag{3.16}$$

The dimension of the manifold matches that of the Euclidean space it is homeomorphic to. In case of the earth (or the unit sphere S^2) this Euclidean space would be $\mathbb{R}^2$, therefore S^2 is a two-dimensional manifold.

Representing distinct points on the manifold can be performed with respect to coordinate frames that are either globally or locally defined [Choset et al., 2005, Chapter 3.5]. A global representation in this regard utilizes a coordinate frame defined on the Euclidean space the manifold is embedded in. A straightforward example is the earth, seen as a two-dimensional manifold that can be embedded in the three-dimensional Euclidean space $\mathbb{R}^3$. If we embed the unit sphere S^2 into $\mathbb{R}^3$ in such a way that its center aligns with the origin of $\mathbb{R}^3$, every point x of the embedded sphere S^2 can be defined as a three-dimensional vector from $\mathbb{R}^3$. Such an embedding is proven to exist for all smooth manifolds and has an upper limit. The Whitney embedding theorem [Whitney, 1944] states that for any smooth manifold M with dimension n, there exists a smooth embedding into the Euclidean space $\mathbb{R}^{2n}$. However, this embedding is not necessarily the minimal one. According to the Whitney embedding theorem, S^2 would be embedded in $\mathbb{R}^4$ where, as previously shown, S^2 can be embedded in the lower-dimensional $\mathbb{R}^3$.

In contrast to the global reference frame constructed by embedding the manifold in a higher-dimensional Euclidean space, a local reference frame for a point $x \in M$ can be constructed by selecting an arbitrary neighborhood of the manifold $U_x \subset M$ with $x \in U_x$. As it holds that U_x locally resembles the Euclidean space $\mathbb{R}^n$, x can be represented as a vector from $\mathbb{R}^n$ with respect

to the local coordinate function $\phi_x : U_x \to \mathbb{R}^n$ (see Eq. (3.14)). Thus, it can be represented with respect to an arbitrary basis $B^n = \{\hat{b}_1, \ldots, \hat{b}_n\}$ or the natural basis vectors $\hat{e}_1, \ldots, \hat{e}_n$ of $\mathbb{R}^n$, respectively.

3.3.1 Paths on Manifolds

We now revisit the question on how to represent a transition from a given configuration q_i to another configuration q_j. As we established the configuration space to be a smooth manifold, moving from an initial configuration q_i to another configuration q_j corresponds to traverse $\mathcal{Q}^n$ along a path [Lee, 2003, p. 608] between these two locations on the manifold. We define a path γ between q_i and q_j on the smooth manifold $\mathcal{Q}^n$ as a smooth function with

$$\gamma_{(q_i,q_j)} : [0, 1] \to \mathcal{Q}^n \tag{3.17}$$

for which it holds that

$$\gamma_{(q_i,q_j)}(0) = q_i \quad \text{and} \quad \gamma_{(q_i,q_j)}(1) = q_j. \tag{3.18}$$

An example for such a path is depicted in Fig. 3.5a. We denote the set of all paths on $\mathcal{Q}^n$ as

$$\Gamma_{\mathcal{Q}^n} = \{\, \gamma_{(q_i,q_j)} \,|\, q_i, q_j \in \mathcal{Q}^n \,\}, \tag{3.19}$$

where $\gamma_{(q_i,q_j)}$ is defined according to Eq. (3.17). If a path from this set exists for arbitrary elements $q_i, q_j \in \mathcal{Q}^n$, the smooth manifold $\mathcal{Q}^n$ is said to be path-connected [Lee, 2003, p. 608], thus it holds that

$$\forall q_i, q_j \in \mathcal{Q}^n : \exists\, \gamma_{(q_i,q_j)} \in \Gamma_{\mathcal{Q}^n}, \tag{3.20}$$

where in the following we assume that $\mathcal{Q}^n$ has this properties. Following a path γ gradually changes the configuration of the entity, effectively encoding the motion from the starting configuration q_i to the destination q_j. From a mathematical point of view, for a path-connected manifold such paths exist per definition for arbitrary pairs $q_i, q_j \in \mathcal{Q}^n$. However, as already established for individual configurations, depending on the scenario, the movement of a robotic system might be obstructed. To account for this we denote a path $\gamma_{(q_i,q_j)}$ to be free if q_i, q_j and all configurations in between are not obstructed [Choset et al., 2005, Chapter 3.2], thus

$$\text{free}_{\circ} : \Gamma_{\mathcal{Q}^n} \to \{\, \mathit{true}, \mathit{false} \,\}, \tag{3.21}$$

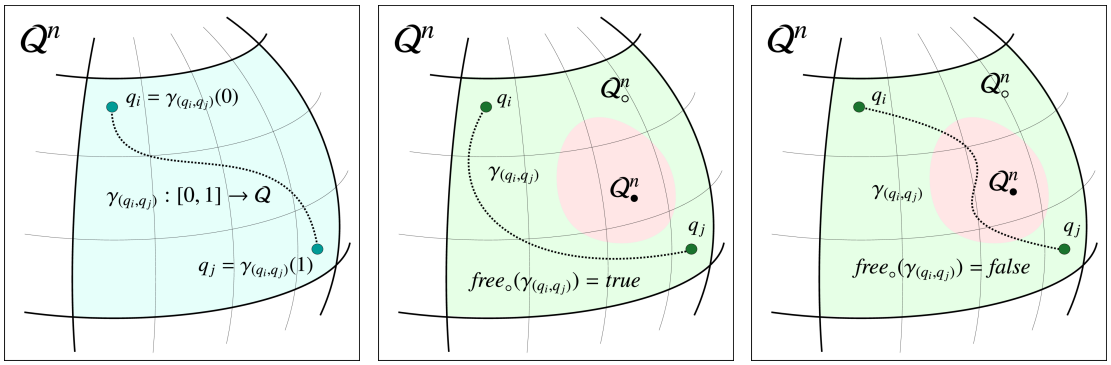
where it holds that

$$\text{free}_{\circ}(\gamma_{(q_i,q_j)}) = \mathit{true} \quad \text{iff} \quad \forall\, t \in [0, 1] : \gamma_{(q_i,q_j)}(t) \in \mathcal{Q}^n_{\circ} \tag{3.22}$$

and accordingly

$$\text{free}_{\circ}(\gamma_{(q_i,q_j)}) = \mathit{false} \quad \text{iff} \quad \exists\, t \in [0, 1] : \gamma_{(q_i,q_j)}(t) \in \mathcal{Q}^n_{\bullet}, \tag{3.23}$$

where $\mathcal{Q}^n_{\circ}$ and $\mathcal{Q}^n_{\bullet}$ are defined with respect to the entity-specific occupancy function ϱ (see Eq. (3.10)). Both cases are depicted in Fig. 3.5b and Fig. 3.5c, respectively. In addition to



(a) Path on $\mathcal{Q}^n$ as the function $\gamma_{(q_i,q_j)}$ that maps from $[0,1]$ to $\mathcal{Q}^n$.

(b) Unobstructed path $\gamma_{(q_i,q_j)}$, where $\text{free}_\circ(\gamma_{(q_i,q_j)})$ is *true* as for all $t \in [0,1]$ it holds that $\gamma_{(q_i,q_j)}(t)$ is in $\mathcal{Q}^n_\circ$.

(c) Obstructed path $\gamma_{(q_i,q_j)}$, where $\text{free}_\circ(\gamma_{(q_i,q_j)})$ is *false* as for at least one $t \in [0,1]$ it holds that $\gamma_{(q_i,q_j)}(t)$ is in $\mathcal{Q}^n_\bullet$.

Figure 3.5: Depiction of paths on the manifold $\mathcal{Q}^n$.

defining whether the entirety of a path is free or obstructed, for practical applications it is often useful to be able to determine what portion of a given path is unobstructed and which portion is obstructed. To that end we define the function

$$\text{free}_\otimes : \Gamma_{\mathcal{Q}^n} \rightarrow [0,1] \tag{3.24}$$

that yields the length of the maximal free portion of a path $\gamma_{(q_i,q_j)} \in \Gamma_{\mathcal{Q}}$ as a real number $t^\otimes$ in the range $[0,1]$ with

$$\text{free}_\otimes(\gamma_{(q_i,q_j)}) = \begin{cases} 0, & \text{if} \quad q_i \in \mathcal{Q}^n_\bullet \\ 1, & \text{if} \quad \text{free}_\circ(\gamma_{(q_i,q_j)}) = true, \\ t^\otimes \in (0,1), & \text{otherwise} \end{cases} \tag{3.25}$$

where in the last case of Eq. (3.25) $t^\otimes \in (0,1)$ denotes the end of the free path, thus

$$\forall t \in [0, t^\otimes] : \gamma_{(q_i,q_j)}(t) \in \mathcal{Q}^n_\circ \quad \text{and} \quad \gamma_{(q_i,q_j)}(t^\otimes + \epsilon) \in \mathcal{Q}^n_\bullet, \tag{3.26}$$

where ϵ denotes a small, non-zero, real-valued number. The first case where $q_i \in \mathcal{Q}^n_\bullet$ refers to situations in which the starting configuration q_i is obstructed. The second case, where $\text{free}_\circ(\gamma_{(q_i,q_j)}) = true$, refers to situations in which the entire path $\gamma_{(q_i,q_j)}$ is unobstructed. The last case refers to situations in which the path is only partially obstructed. Here $t^\otimes$ denotes the last point of the path that is unobstructed, i.e., it holds that $\gamma_{(q_i,q_j)}(t^\otimes) = q^\otimes$ with $q^\otimes \in \mathcal{Q}^n_\circ$. As the agent was unable to follow the path any further, the next configuration is obstructed, i.e., it holds that $\gamma_{(q_i,q_j)}(t^\otimes + \epsilon) = q^\ominus$ with $q^\ominus \in \mathcal{Q}^n_\bullet$. Here ϵ corresponds to a small, non-zero, real-valued delta that indicates a transition along the path and, thus, corresponds to the change from the unobstructed configuration given by $t^\otimes$ to an obstructed one. However, while we know that the configuration directly after $t^\otimes$ is obstructed, we can not make any statement regarding the remainder of the path $\gamma_{(q_i,q_j)}$, i.e., the section between $q^\otimes = \gamma_{(q_i,q_j)}(t^\otimes + \epsilon)$ and $q_j = \gamma_{(q_i,q_j)}(1)$. A visualization of such a partial obstructed path is given in Figure 3.6.

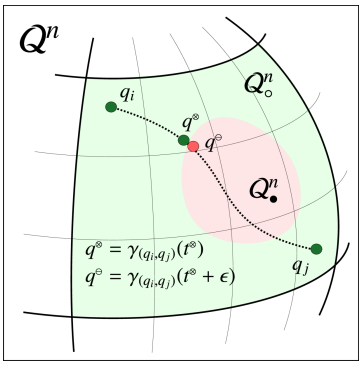


Figure 3.6: Partially obstructed path on the configuration space. An agent starts at the unobstructed configuration q_i and moves along $\gamma_{(q_i,q_j)}$ until it reaches the configuration $q^\otimes = \gamma_{(q_i,q_j)}(t^\otimes)$ that marks the last unobstructed configuration on the path from q_i to q_j. The next configuration $q^\ominus = \gamma_{(q_i,q_j)}(t^\otimes) + \epsilon$ is obstructed where ϵ denotes a small, non-zero, real-valued delta. The parameter $t^\otimes \in (0, 1)$ corresponds to $q^\otimes$, the last unobstructed configuration on the path. Though other unobstructed configurations on $\gamma_{(q_i,q_j)}$ might exist, they are separated from $q^\otimes$ by at least the obstructed configuration $q^\ominus$.

3.3.2 Parameterization of Paths

So far, we established the relation between changes in the configuration of an entity and the topology of the configuration space by expressing configuration changes as paths on an n-dimensional manifold. However, we have yet to parameterize these paths to be able to relate them to motor controls. The next step towards this goal is to establish a relation between these paths and the local manifold structure, i.e., relate changes on the n-dimensional manifold $\mathcal{Q}^n$ to the associated vector space $\mathbb{R}^n$. A general approach for such a mapping is proposed in [Hertzberg et al., 2011] by defining two encapsulating operators $\boxplus$ and $\boxminus$ (denoted 'boxplus' and 'boxminus', respectively with

$$\boxplus : \mathcal{Q}^n \times \mathbb{R}^n \to \mathcal{Q}^n \quad \text{and} \tag{3.27}$$

$$\boxminus : \mathcal{Q}^n \times \mathcal{Q}^n \to \mathbb{R}^n. \tag{3.28}$$

The $\boxplus$-operator applies a small change as an element from a local vector space $d \in \mathbb{R}^n$ to a given manifold element q_i and calculates the resulting new element q_j as

$$q_i \boxplus d = q_j \quad \text{with} \quad q_i, q_j \in \mathcal{Q}^n \quad \text{and} \quad d \in \mathbb{R}^n. \tag{3.29}$$

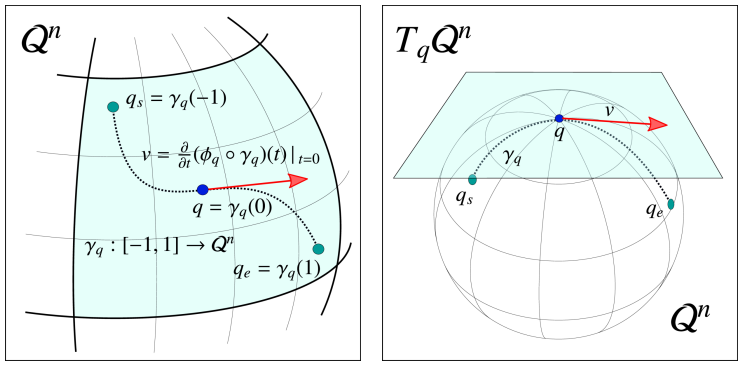
The $\boxminus$-operator calculates the delta $d \in \mathbb{R}^n$ between two given elements q_i, q_j on the manifold, thus

$$q_j \boxminus q_i = d \quad \text{with} \quad q_i, q_j \in \mathcal{Q}^n \quad \text{and} \quad d \in \mathbb{R}^n, \tag{3.30}$$

where it consequently holds that

$$q_i \boxplus d = q_j \quad \text{iff} \quad q_j \boxminus q_i = d. \tag{3.31}$$

These operators encapsulate calculations on the manifold by representing configuration changes in the local vector space. This allows algorithms, originally defined to work on Euclidean vector



(a) Instance of a tangent vector v at $q \in Q^n$ as derivative of the curve γ_q passing through q at $t = 0$. The curve is defined as the function $\gamma_q : [-1, 1] \to Q^n$ where q_s denotes the start point $q_s = \gamma_q(-1)$ and q_s its end point $q_e = \gamma_q(1)$.

(b) Tangent space at $q \in Q^n$ as the set of all tangent vectors at q, i.e., all tangent vectors of all possible curves passing through q. Example tangent vector at $q \in Q^n$ given as $v \in T_qQ^n$. Figure based on [Bloch et al., 2007, p. 67, Figure 2.2.4]

Figure 3.7: Visualization of tangent vectors and tangent spaces on the smooth manifold Q^n.

spaces, to work on arbitrary smooth manifolds that support these two operators. For a more in-depth description we refer to [Hertzberg et al., 2011].

Next, we look at the relation between the local vector space $\mathbb{R}^n$ of a given manifold Q^n and the previously defined paths, i.e., the relation between these paths and the delta $d \in \mathbb{R}^n$. To be able to relate paths on smooth manifolds to elements from a local vector space and ultimately relate these to motor controls, we need the notion of tangent vectors and tangent spaces on smooth manifolds.

3.4 Tangent Spaces

The notation and examples within this section are based on [Bloch et al., 2007, Chapter 2.2 and 2.9], [Choset et al., 2005, Chapter 12.4.1], and [Lee, 2003, Chapter 3 and 13].

Let q be a point on the arbitrary n-dimensional smooth manifold Q^n. We select a chart (U_q, ϕ_q) on Q^n so that $U_q \ni q$ and ϕ_q denotes the coordinate function (see Eq. (3.14)). Now let $\gamma_q(t)$ be a smooth curve passing through q with

$$\gamma_q : [-1, 1] \to Q^n \quad \text{and} \quad \gamma_q(0) = q. \tag{3.32}$$

Intuitively, this curve can be seen as describing the trajectory of an object moving along Q^n and passing through q at $t = 0$ as visualized in Fig. 3.7a. We can express γ_q with respect to $\mathbb{R}^n$ by composition of γ_q and ϕ_q. This yields

$$(\phi_q \circ \gamma_q) : [-1, 1] \to \mathbb{R}^n. \tag{3.33}$$

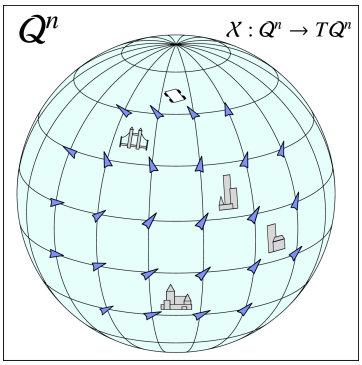


Figure 3.8: Vector field $\mathcal{X} : Q^n \to TQ^n$ on the smooth manifold Q^n. Depiction of tangent vectors as elements from TQ^n at distinct points on Q^n.

Taking the derivative of this function at $t = 0$ yields

$$v_q = \frac{d}{dt}(\phi_q \circ \gamma_q)(t)\,|_{t=0}, \tag{3.34}$$

the tangent vector corresponding to γ_q at q as an element from $\mathbb{R}^n$ describing the rate of change, i.e., the direction and speed of the trajectory at that point [Bloch et al., 2007, p.66]. The set of all tangent vectors at a given point $q \in Q^n$ is given by the tangent space T_qQ^n and it is a real vector space with the same dimension as the manifold itself, which is visualized in Fig. 3.7b.

Another important concept that is related to the definition of tangent spaces is that of the tangent bundles and vector fields. The tangent bundle TQ^n of a smooth manifold Q^n is the collection of all tangent spaces T_qQ^n from all points $q \in Q^n$, or alternatively, the set of all possible tangent vectors at all points on the manifold. Formally, TQ^n is defined as the disjoint union [Lee, 2003, p. 65] of all tangent spaces, thus

$$TQ^n = \bigsqcup_{q \in Q^n} T_qQ^n = \{\,(q, v)\,|\,q \in Q^n, v \in T_qQ^n\,\}. \tag{3.35}$$

Assigning a tangent vector to each point of the manifold, yields a vector field that is defined as the function

$$\mathcal{X} : Q^n \to TQ^n \tag{3.36}$$

that maps from the manifold into the tangent bundle, therefore it is a function that assigns a specific tangent vector from its tangent space T_qQ^n to every point $q \in Q^n$ [Bloch et al., 2007, p. 68]. A visual example is global weather map depicting wind directions and speeds. The sphere S^2 (which is a suitable approximation of the earth) is a smooth two-dimensional manifold with local tangent spaces that are homeomorphic to $\mathbb{R}^2$. Thus, wind direction and speed (parallel to the surface) can be expressed by tangent vectors $v \in \mathbb{R}^2$ as depicted in Fig. 3.8.

If the tangent space of a smooth manifold is equipped with an inner product

$$\otimes : T_qQ^n \times T_qQ^n \to \mathbb{R} \quad \text{with} \quad q \in Q^n, \tag{3.37}$$

a function that assigns a real number to a pair of tangent vectors, the manifold is called a Riemannian manifold [Choset et al., 2005, pp. 428–429]. This inner product is required to be able to define distances and angles between vectors and thus ultimately allows to calculate lengths of curves on the manifold. A common example for such an inner product is the dot product in Euclidean space $\mathbb{R}^n$ [Lee, 2003, pp. 327–328]. For more examples and a more thorough discussion of this subject, we refer to [Lee, 2003, Chapter 13].

Being able to calculate the lengths of curves allows us to determine the length of any given path $\gamma_{(q_i,q_j)}$ (see Eq. (3.17)) between two points $q_i, q_j \in \mathcal{Q}^n$, defined as

$$\text{length} : \Gamma_{\mathcal{Q}^n} \to \mathbb{R}^+. \tag{3.38}$$

This allows us to calculate the length of the free portion of a given path with respect to Eqs. (3.24) to (3.26). To facilitate this calculation, we consider that for a path-connected Riemannian manifold $\mathcal{Q}^n$, given two arbitrary points $q_i, q_j \in \mathcal{Q}^n$, there exists a locally shortest path between those points that is called a geodesic [Choset et al., 2005, p. 428] where we we denote a geodesic between q_i, q_j as $\vec{\gamma}_{(q_i,q_j)}$.

$$\vec{\gamma}_{(q_i,q_j)} = \underset{\gamma_{(q_i,q_j)} \in \Gamma_{\mathcal{Q}^n}}{\arg\min}\ \text{length}(\gamma_{(q_i,q_j)}) \tag{3.39}$$

In the analogy of paths describing trajectories on $\mathcal{Q}^n$, a geodesic corresponds to a trajectory that describes an object moving from q_i to q_j while maintaining constant velocity, i.e., constant direction and speed [Bloch et al., 2007, p. 110]. If the velocity along $\vec{\gamma}_{(q_i,q_j)}$ is constant, this implies that the same holds for the tangent vectors at each point $q_t \in \mathcal{Q}^n$ that $\vec{\gamma}_{(q_i,q_j)}$ passes through. However, if the tangent vectors are constant along the entirety of $\vec{\gamma}_{(q_i,q_j)}$, each geodesic can be parameterized by its starting point q_i and the tangent vector corresponding to $\vec{\gamma}_{(q_i,q_j)}$, i.e., the tangent vector $v_{\gamma_{(q_i,q_j)}} \in T_{q_i}\mathcal{Q}^n$ from that local tangent space alone. Accordingly, the length of the geodesic $\vec{\gamma}_{(q_i,q_j)}$ is proportional to this tangent vectors, thus

$$\text{length}(\vec{\gamma}_{(q_i,q_j)}) \propto \|v_{\gamma_{(q_i,q_j)}}\| \quad \text{with} \quad v_{\gamma_{(q_i,q_j)}} \in T_{q_i}\mathcal{Q}^n. \tag{3.40}$$

As the tangent space is a real vector space with the same dimension as the manifold itself, we can represent elements from the tangent space as n-tuples

$$v_{\gamma_{(q_i,q_j)}} = r_1\hat{b}_1 + r_2\hat{b}_2 + \cdots + r_n\hat{b}_n \tag{3.41}$$

with $r_1, \cdots, r_n \in \mathbb{R}$ and $\hat{b}_1, \cdots, \hat{b}_n \in B^n$, where B^n denotes an arbitrary basis for the tangent space $T_{q_i}\mathcal{Q}^n$, i.e., a set of elements from $\mathbb{R}^n$. If the configuration space of an entity is a path-connected Riemannian manifold, a change from configuration q_i to q_j can therefore be parametrized by a tangent vector from the tangent space at $T_{q_i}\mathcal{Q}^n$, i.e., an element from the real vector space $\mathbb{R}^n$. Naturally, the number of independent basis vectors of this vector space corresponds to the dimension of the manifold. The basis vectors of the tangent space can thus be seen as elementary building blocks for any configuration change the entity can perform. Thus. we can relate the previously discussed operators $\boxplus$ and $\boxminus$ (Eqs. (3.27) and (3.28)) to the tangent

space of such a manifold by substituting the generic $\mathbb{R}^n$ with $T_q\mathcal{Q}^n$, yielding an explicit definition with

$$\boxplus_{[T_q\mathcal{Q}^n]} : \mathcal{Q}^n \times T_q\mathcal{Q}^n \to \mathcal{Q}^n \quad \text{and} \tag{3.42}$$

$$\boxminus_{[T_q\mathcal{Q}^n]} : \mathcal{Q}^n \times \mathcal{Q}^n \to T_q\mathcal{Q}^n. \tag{3.43}$$

Here Eq. (3.42) corresponds to traversing the manifold along a geodesic, defined by a tangent vector from the tangent space of the current configuration with the endpoint being the new configuration. Notably, the tangent space $T_q\mathcal{Q}^n$ is associated with the starting configuration $q \in \mathcal{Q}^n$. Consequently, Eq. (3.43) describes calculating the geodesic between two configurations $q_i, q_j \in \mathcal{Q}^n$ and returning the corresponding tangent vector from the tangent space at q_i. For configuration spaces that are Riemannian manifolds, motor controls can thus be related to tangent vectors and ultimately be expressed as elements from $\mathbb{R}^n$.

The drawback of this approach is that both operations are manifold specific. This means that in order to utilize Eqs. (3.42) and (3.43), we have to know the properties of the manifold we are working with, i.e.,how to calculate and follow geodesics and how to calculate tangent vectors where, as established in Section 2.3.3, manifold learning, i.e., detecting the properties of an unknown manifold is a challenging task. In order to overcome this difficulty, we with a certain type of smooth manifolds, denoted as Lie groups.

3.5 Lie Groups

The notation and examples within this section are based on [Stillwell, 2008, Chapter 2.1], [Lee, 2003, Chapter 7], [Choset et al., 2005, Chapters 3.5.1 and 3.6], and [Selig, 2007, Chapter 4.4]

Lie groups are special kinds of manifolds as they combine two mathematical concepts. On the one hand, they are smooth manifolds, on the other hand, they are groups [Stillwell, 2008, Chapter 2.1]. A group is defined as a tuple $(G, \bullet)$ of a set G together with a group operation

$$\bullet : G \times G \to G \tag{3.44}$$

that satisfies the group axioms

$$\begin{aligned}
&\text{closure} && \forall\, a, b \in G && : a \bullet b \in G, && (3.45)\\
&\text{associativity} && \forall\, a, b, c \in G && : (a \bullet b) \bullet c = a, \bullet (b \bullet c), && (3.46)\\
&\text{identity element} && \exists\, id \in G && : id \bullet a = a \bullet id = a, && (3.47)\\
&\text{inverse element} && \forall\, a \in G && : \exists\, a^{-1} \in G : a^{-1} \bullet a = a \bullet a^{-1} = id. && (3.48)
\end{aligned}$$

Though associativity has to hold for all groups, commutativity with

$$\forall\, a, b \in G : a \bullet b = b \bullet a \tag{3.49}$$

does not necessarily. Groups for which the group operation $\bullet$ is commutative are called abelian or commutative groups. In obvious cases, we omit the group operation symbol $\bullet$, thus $g_i \bullet g_j$ will

be equivalent to $g_i g_j$. In the remainder of this thesis we look at special types of Lie groups called matrix Lie groups. More precisely, we look at matrix Lie groups with entries from $\mathbb{R}$. The most general one is the general linear group $GL(n, \mathbb{R})$, which consists of all invertible $n \times n$ matrices with the matrix multiplication as group operation [Lee, 2003, p. 151], thus

$$GL(n, \mathbb{R}) = \{ A \in \mathbb{R}^{n \times n} \mid \det(A) \neq 0 \}. \tag{3.50}$$

For matrix Lie groups, the identity element id corresponds to the $n \times n$ identity matrix

$$I_{n \times n} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}. \tag{3.51}$$

If not stated otherwise, in the remainder of this thesis $\mathcal{Q}^n$ denotes a matrix Lie group with group elements $q \in \mathcal{Q}^n$ and identity element id. Within the context of this thesis we consider matrix Lie groups that describe translation and rotation in n-dimensional Euclidean space. Notable matrix Lie groups in this regard are $SO(2)$, $SO(3)$ and $SO(n)$ describing rotation in 2-, 3-, and n-dimensional Euclidean space where $SE(2)$, $SE(3)$ and $SE(n)$ describe both translation and rotation in 2, 3, and n dimensions respectively [Choset et al., 2005, pp. 60–69]. An overview of these groups and their properties is given in Appendix A.

Endowing a smooth manifold with a group structure has interesting implications with respect to its elements. On the one hand, each element $q \in \mathcal{Q}^n$ naturally represents a location on the manifold. On the other hand, however, each $q \in \mathcal{Q}^n$ encodes a smooth translation, i.e., a smooth movement along the manifold $\mathcal{Q}^n$. Thus, given two elements $q_i, q_j \in \mathcal{Q}^n$, the transformation from q_i to q_j is constructed as $q_{ij} = q_i^{-1} q_j$, where due to the closure of $\mathcal{Q}^n$ naturally it holds that $q_{ij} \in \mathcal{Q}^n$. We define $\forall q, x \in \mathcal{Q}^n$

$$l_q : \mathcal{Q}^n \to \mathcal{Q}^n \quad \text{with} \quad l_q(x) = q \bullet x \quad \text{and} \tag{3.52}$$

$$r_q : \mathcal{Q}^n \to \mathcal{Q}^n \quad \text{with} \quad r_q(x) = x \bullet q \tag{3.53}$$

where l_q is the left translation and r_q is the right translation, respectively [Lee, 2003, p.151]. As the group operation $\bullet$ is smooth per definition, both l_q and r_q are, too. Thus, left and right translation allow us to move group elements smoothly across the manifold. Applying l_g and r_g to the identity element $id \in \mathcal{Q}^n$ yields

$$l_q(id) = q \bullet id = q \quad \text{and} \tag{3.54}$$

$$r_q(id) = id \bullet q = q. \tag{3.55}$$

This implies that each element $q \in \mathcal{Q}^n$ does not only encode a smooth translation for arbitrary other elements, but that it itself encodes the smooth translation from the identity element of the group to itself. Thus, each group element $q \in \mathcal{Q}^n$ represents a path from id to q and, moreover, this path is a geodesic $\vec{\gamma}_{(id,q)}$ on $\mathcal{Q}^n$. Left and right translation can therefore be seen as the ability to move paths smoothly along other paths on the manifold.

As Lie groups are smooth manifolds, for each element $q \in \mathcal{Q}^n$ the tangent space $T_q\mathcal{Q}^n$ exists and can be interpreted as the set of velocities of all curves passing through q. In the case of Lie groups the special case holds that every tangent space $T_q\mathcal{Q}^n$ is isomorphic to the tangent space at the identity of the group $T_{id}\mathcal{Q}^n$. This means that the tangent space $T_q\mathcal{Q}^n$ for all $q \in \mathcal{Q}^n$ looks similar to the tangent space $T_{id}\mathcal{Q}^n$, and thus tangent vectors on Lie groups can be uniformly expressed with respect to the tangent space at the group's identity $T_{id}\mathcal{Q}^n$. This tangent space isomorphism implies that the structure of the manifold is completely described by the tangent vectors from the tangent space at the identity of the group, as we can translate them to any other tangent space without changing their properties in any way. As tangent vectors are invariant to translation (either with respect to l_q or r_q) this allows us to draw a connection between the tangent space at the identity $T_{id}\mathcal{Q}^n$ and vector fields on $\mathcal{Q}^n$ (see Eq. (3.36)). As for Lie groups all tangent spaces are isomorphic to the tangent space at the group identity, each tangent vector from $T_{id}\mathcal{Q}^n$ directly corresponds to a distinct vector field, as it can be 'moved' across the entire manifold without changing the tangent vector's properties [Selig, 2007, Chapter 4.4]. We now take a closer look at the tangent space of a Lie group, which is called the Lie algebra.

3.6 Lie Algebras

The notation and examples within this section are based on [Lee, 2003, Chapter 8], [Choset et al., 2005, Chapter 12.1.3], and [Stillwell, 2008, Chapters 4 and 7].

The tangent space $T_{id}\mathcal{Q}^n$ of a Lie group $\mathcal{Q}^n$ is a vector space called the Lie algebra $\mathfrak{Q}^n$, where elements from the Lie algebra $\mathfrak{Q}^n$ are denoted by $\mathfrak{q}$ [Lee, 2003, pp. 189ff.].[3] The basis vectors of the Lie algebra are called generators, denoted by $\hat{\mathfrak{g}}_1, \cdots, \hat{\mathfrak{g}}_n$, and, as the Lie algebra is a real vector space, every tangent vector $\mathfrak{q} \in \mathfrak{Q}^n$ can be expressed by a linear combination of these generators. Thus, for all $\mathfrak{q} \in \mathfrak{Q}^n$ there exists

$$\mathfrak{q} = r_1\hat{\mathfrak{g}}_1 + r_2\hat{\mathfrak{g}}_2 + \cdots + r_n\hat{\mathfrak{g}}_n \quad \text{with} \quad r_1, \cdots, r_n \in \mathbb{R}^n. \tag{3.56}$$

Though we refer to the generators as basis vectors, it has to be noted that the Lie algebra of a matrix Lie group has the same matrix structure as the group, thus the 'vectors' are actually $n \times n$ matrices. In the remainder of this thesis, we abbreviate tangent vectors by omitting their basis vector components, representing them as vectors from $\mathbb{R}^n$, thus

$$\begin{aligned} &\mathfrak{q} = r_1\hat{\mathfrak{g}}_1 + r_2\hat{\mathfrak{g}}_2 + \cdots + r_n\hat{\mathfrak{g}}_n && \text{corresponds to} \\ &\mathfrak{q} = [\, r_1\hat{\mathfrak{g}}_1, r_2\hat{\mathfrak{g}}_2, \cdots, r_n\hat{\mathfrak{g}}_n \,]^T && \text{and in a shortened notation to} \\ &\mathfrak{q} = [\, r_1, r_2, \cdots, r_n \,]^T. \end{aligned} \tag{3.57}$$

As $\mathfrak{Q}^n$ and $T_{id}\mathcal{Q}^n$ both denote the tangent space at the identity of the Lie group, we use these notations interchangeably, depending on the current context. In addition to being a real vector

[3] In literature, a Lie group and its corresponding algebra are typically denoted by G and $\mathfrak{g}$. We chose to stick with $\mathcal{Q}^n$ and $\mathfrak{Q}^n$ to maintain consistency and to emphasize the utilization of both concepts with respect to representing the configuration space of a mobile entity.

space, the Lie algebra is equipped with a binary operation

$$[\,\cdot\,,\,\cdot\,] : \mathfrak{Q}^n \times \mathfrak{Q}^n \to \mathfrak{Q}^n \tag{3.58}$$

called the Lie bracket [Lee, 2003, pp. 185–189] that satisfies the axioms

$$\text{antisymmetry with} \quad \forall\, \mathfrak{a}, \mathfrak{b} \in \mathfrak{Q}^n \quad : \; [\mathfrak{a}, \mathfrak{b}] = -[\mathfrak{b}, \mathfrak{a}] \quad \text{and} \tag{3.59}$$

$$\text{Jacobi identity with} \quad \forall\, \mathfrak{a}, \mathfrak{b}, \mathfrak{c} \in \mathfrak{Q}^n \quad : \; [\mathfrak{a}, [\mathfrak{b}, \mathfrak{c}]] + [\mathfrak{b}, [\mathfrak{c}, \mathfrak{a}]] + [\mathfrak{c}, [\mathfrak{a}, \mathfrak{b}]] = 0. \tag{3.60}$$

For matrix Lie groups the Lie bracket is defined as the commutator

$$[\mathfrak{a}, \mathfrak{b}] = \mathfrak{a}\mathfrak{b} - \mathfrak{b}\mathfrak{a} \quad \text{with} \quad \mathfrak{a}, \mathfrak{b} \in \mathfrak{Q}^n \tag{3.61}$$

where $\mathfrak{a}\mathfrak{b}$ and $\mathfrak{b}\mathfrak{a}$ denote matrix multiplication. To understand what this operator does, we have to remember the correspondence between elements of the tangent space and vector fields on the manifold, as any given tangent space element from $\mathfrak{Q}^n$ corresponds to a certain vector field on $\mathcal{Q}^n$. Applied to two elements $\mathfrak{a}, \mathfrak{b} \in \mathfrak{Q}^n$, the Lie bracket $[\mathfrak{a}, \mathfrak{b}]$ answers the question: "What is the difference between 'follow $\mathfrak{a}$ and then $\mathfrak{b}$' and 'follow $\mathfrak{b}$ and then $\mathfrak{a}$'?". If the two vector fields commute, the effects of $\mathfrak{a}\mathfrak{b}$ and $\mathfrak{b}\mathfrak{a}$ are the same, thus the answer to the above question would obviously be "none", i.e., $[\mathfrak{a}, \mathfrak{b}] = 0_{n\times n}$. However, if $[\mathfrak{a}, \mathfrak{b}]$ is not zero, it yields a new vector field that describes the displacement one can achieve by alternately following $\mathfrak{a}$ and $\mathfrak{b}$. [Choset et al., 2005, Chapter 12.1.3]

Lie groups and their corresponding Lie algebras are related, as there exists a mapping from the Lie algebra to the Lie group. It allows one to map elements from the tangent space at the identity to group elements [Stillwell, 2008, Chapter 4]. It is given by

$$\exp : T_{id}\mathcal{Q}^n \to \mathcal{Q}^n. \tag{3.62}$$

The inverse allows to map elements from the Lie group to the tangent space at the identity. [Stillwell, 2008, Chapter 7]. It is given as

$$\log : \mathcal{Q}^n \to T_{id}\mathcal{Q}^n. \tag{3.63}$$

It holds that Eq. (3.62) is guaranteed to succeed for all elements in the tangent space, i.e., all Lie group elements can be created by applying the exponential map to a corresponding element from the Lie algebra, thus

$$\forall q \in \mathcal{Q}^n \; : \; \exists\, \mathfrak{q} \in T_{id}\mathcal{Q}^n \; : \; \exp(\mathfrak{q}) = q. \tag{3.64}$$

However, for Eq. (3.63) the same does not hold as there exist cases where the logarithm of a group element $q \in \mathcal{Q}^n$ is not unique. An example is the following Lie algebra element $\mathfrak{q}$ and the corresponding group element q, calculated by applying the exponential map with

$$\mathfrak{q} = \begin{bmatrix} 0 & -\pi \\ \pi & 0 \end{bmatrix}, \quad \exp(\mathfrak{q}) = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}, \tag{3.65}$$

where the geometric interpretation of these matrices corresponds to a positive 180° rotation in the x/y-plane. However, we can not calculate the inverse as the matrix logarithm of q does not

provide a unique solution, which, from a geometric point of view, corresponds to the ambiguity of being able to move from id to q either by applying a positive π or a negative $-\pi$ rotation. For matrix Lie groups, the exponential map corresponds to the matrix exponential where the inverse transformation corresponds to the matrix logarithm.

The exponential map and the matrix logarithm relate tangent space elements $\mathfrak{q} \in T_{id}\mathcal{Q}^n$ to group elements $q \in \mathcal{Q}^n$. As it holds that elements $q \in \mathcal{Q}^n$ correspond to geodesics $\vec{\gamma}_{(e,q)}$ on $\mathcal{Q}^n$, there exists a direct correspondence between vectors in $T_{id}\mathcal{Q}^n$ and these special curves on the manifold. Given that the exponential map and its inverse are explicitly defined for matrix Lie groups, we can apply these functions to Eqs. (3.42) and (3.43), yielding

$$\boxplus_{[\text{Lie}]} : \mathcal{Q}^n \times T_{id}\mathcal{Q}^n \rightarrow \mathcal{Q}^n, \tag{3.66}$$

$$\boxminus_{[\text{Lie}]} : \mathcal{Q}^n \times \mathcal{Q}^n \rightarrow T_{id}\mathcal{Q}^n \tag{3.67}$$

with the $\boxplus_{[\text{Lie}]}$-operator on matrix Lie groups given by

$$q_i \boxplus_{[\text{Lie}]} \mathfrak{q} = q_i \bullet \exp(\mathfrak{q}) = q_i \bullet q = q_j \quad \text{with} \quad q_i, q_j, q \in \mathcal{Q}^n, \mathfrak{q} \in T_{id}\mathcal{Q}^n \tag{3.68}$$

and the $\boxminus_{[\text{Lie}]}$-operator on matrix Lie groups given by

$$q_j \boxminus_{[\text{Lie}]} q_i = \log(q_i^{-1} \bullet q_j) = \log(q) = \mathfrak{q} \quad \text{with} \quad q_i, q_j, q \in \mathcal{Q}^n, \mathfrak{q} \in T_{id}\mathcal{Q}^n, \tag{3.69}$$

which corresponds to the Lie group specific implementation of the $\boxplus$ and $\boxminus$ operators proposed in [Hertzberg et al., 2011].

As previously stated (see Eqs. (3.56) and (3.57)), it holds that $\mathfrak{d}$, as an element of the tangent space, can be expressed by a linear combination of Lie algebra basis vectors. For the matrix Lie groups we are interested in, these basis vectors are explicitly given. If we utilize matrix Lie groups for representing the configuration space of a mobile entity, the $\boxplus_{[\text{Lie}]}$- and $\boxminus_{[\text{Lie}]}$-operators, expressed in terms of the matrix exponential and its inverse, therefore allow us to encode configuration changes on the manifold as real vectors, providing means of applying controls to a given configuration utilizing Eq. (3.68) or calculate the controls necessary for performing a certain change using Eq. (3.69). A list of prominent Lie groups as well as their generators is given in Appendix A.

3.7 Holonomic and Nonholonomic Systems

The notation and examples from this section are based on [Choset et al., 2005, Chapter 12] and [Siegwart and Nourbakhsh, 2004, Chapters 2.3.2.3 and 3.4.2].

At the end of Section 3.6 we established the relation between vectors from $\mathbb{R}^n$ and the matrix Lie group based configuration space $\mathcal{Q}^n$, facilitated by expressing the $\boxplus_{[\text{Lie}]}$- and $\boxminus_{[\text{Lie}]}$-operators with respect to the matrix logarithm and its inverse (see Eqs. (3.66) and (3.67)). However, we have yet to establish a formal relation between actual actuator controls and the elements of the tangent space of the configuration manifold.

For now, we define an actuator to be a component that is able to provide a control $c \in \mathbb{R}^m$, where m corresponds to the number of independently controllable parameters, i.e., to the DoF. It

holds that the tangent space at the identity $T_{id}\mathcal{Q}^n$, i.e., the Lie algebra $\mathfrak{Q}$, is a real vector space of dimension n, where n denotes the dimension of the manifold $\mathcal{Q}^n$. To model the impact executing controls has on the configuration, we define the mapping from controls $c \in \mathbb{R}^m$ to tangent vectors $\mathfrak{q} \in \mathfrak{Q}^n$ as the linear map

$$\tau : \mathbb{R}^m \to T_{id}\mathcal{Q}^n \quad \Longrightarrow \quad \tau : \mathbb{R}^m \to \mathbb{R}^n. \tag{3.70}$$

Thus, $\forall c \in \mathbb{R}^m$ there exists a corresponding $\mathfrak{q}_c \in \mathcal{Q}^n$ given by $\mathfrak{q}_c = \tau(c)$ with

$$\mathfrak{q}_c = [\, \tau_1(c)\, \hat{\mathfrak{g}}_1,\, \tau_2(c)\, \hat{\mathfrak{g}}_2,\, \cdots,\, \tau_n(c)\, \hat{\mathfrak{g}}_n \,]^T \tag{3.71}$$

where $\tau_i(q)\, \hat{\mathfrak{g}}_i$ denotes the component-wise mapping of $c \in \mathbb{R}^m$ to the basis vector, i.e., the Lie algebra generator $\hat{\mathfrak{g}}_i \in T_{id}\mathcal{Q}^n$. Consequently, it holds that each component $\tau_i(c)$ corresponds to a $r_i \in \mathbb{R}$. According to the shortened notation (see Eq. (3.57)), this yields

$$\begin{aligned} \mathfrak{q}_c &= [\, \tau_1(c),\, \tau_2(c),\, \cdots,\, \tau_n(c) \,]^T \\ &= [\, r_1,\, r_2\ \cdots,\, r_n \,]^T \end{aligned} \tag{3.72}$$

An important question that arises, is how the degree of freedom of the actuator and the mapping between control vectors and the Lie algebra generators, influences the entity's capability to move from one configuration to another. In this regard we give a brief overview of the concepts of holonomic and nonholonomic systems.

A system is holonomic if the number of controllable degrees of freedom matches the number of degrees of freedom of the configuration space [Siegwart and Nourbakhsh, 2004, Chapter 3.4.2]. Let the dimension of controls $c \in \mathbb{R}^m$ and the dimension of the manifold coincide and let τ be a bijective linear map. Then it holds that

$$\forall \mathfrak{q} \in T_{id}\mathcal{Q}^n : \exists\, c \in \mathbb{R}^m : \mathfrak{q} = \tau(c). \tag{3.73}$$

As for every $q \in \mathcal{Q}^n$ there exists a corresponding element from $T_{id}\mathcal{Q}^n$ (see Eq. (3.64)), it consequently holds that

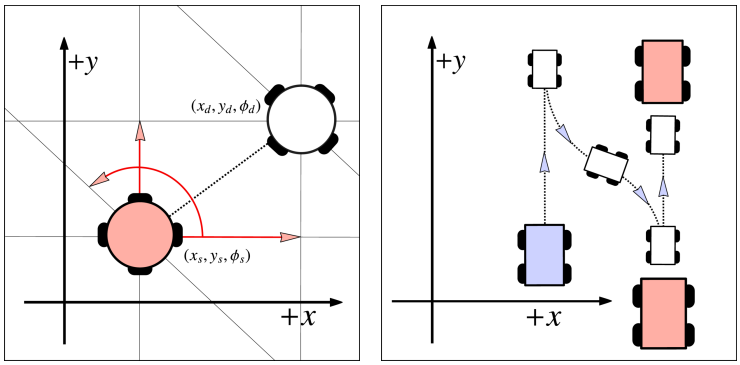
$$\forall q \in \mathcal{Q}^n : \exists\, c \in \mathbb{R}^m : q = \exp(\tau(c)). \tag{3.74}$$

As the transformation between any two group elements $q_i, q_j \in \mathcal{Q}^n$ is given by $q_{ij} \in \mathcal{Q}^n$ with $q_i^{-1} q_j$, Eq. (3.74) implies

$$\forall q_i, q_j \in \mathcal{Q}^n : \exists\, c \in \mathbb{R}^m : q_{ij} = \exp(\tau(c)) \quad \text{where} \quad q_{ij} = q_i^{-1} q_j, \tag{3.75}$$

i.e., for any two configurations $q_i, q_j \in \mathcal{Q}^n$ there exists a $c \in \mathbb{R}^m$ that encodes the geodesic from q_i to q_j. Thus the entity can traverse from q_i to q_j by following said geodesic where an example for a holonomic vacuum cleaning robot is depicted in Fig. 3.9a[4]. If the number of controllable degrees of freedom is lower than the number of degrees of freedom of its configuration space, the system is nonholonomic [Choset et al., 2005, Chapter 12]. Let the dimension of controls

[4] To keep the example simple we assume that the corresponding geodesic is free of any obstruction and the entity can execute any $c \in \mathbb{R}^m$



(a) The holonomic vaccuum cleaning robot can directly move from (x_s, y_s, ϕ_s) to (x_e, y_e, ϕ_e) by combining a translation into x- and y-directions as well as applying a body-fixed rotation. This combined motion corresponds to the control associated to the geodesic from the starting location to the destination.

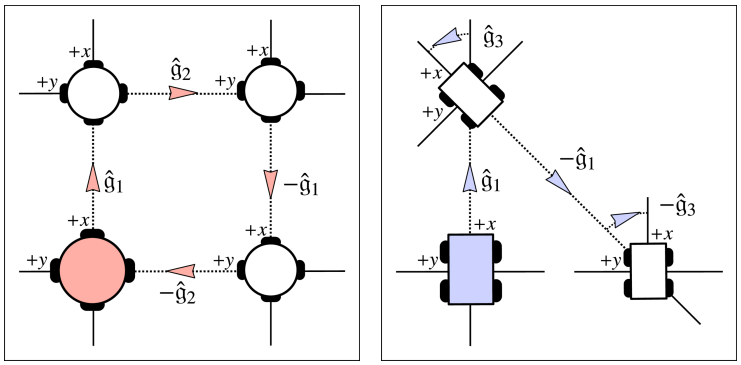
(b) The nonholonomic car-like robot can only move forward, backward and on circular arcs but not directly sideways. To reach the parking space it has to execute a sequence of four different controls.

Figure 3.9: Comparison of a holonomic vaccuum cleaning robot and a car-like nonholonomic system.

$c \in \mathbb{R}^m$ be lower than the dimension of the manifold $\mathcal{Q}^n$. As $m < n$ the mapping τ naturally can not be bijective. In addition it is not determined whether or not τ is surjective either. This implies that there exist some $q \in \mathcal{Q}^n$ for which no corresponding $c \in \mathbb{R}^m$ exist that satisfy Eq. (3.74) and consequently Eq. (3.75) does not hold anymore either. This means that the existence of a control that corresponds to the transformation between two given configurations $q_i, q_j \in \mathcal{Q}^n$ is not guaranteed. The question is, how it is still possible to access the entirety of the configuration space, despite the aforementioned constraints. A prime example for such a nonholonomic system is a car. A car can move forward and backward on a straight line and in addition move along a curved path, where the curvature is determined by the position of the steering wheel.[5] However, a car can not move directly sideways. Nevertheless it is possible to generate such a sideways motion, by combining forward, backward and turning motions (parallel parking) as depicted in Fig. 3.9b. Though the number of controllable degrees of freedom of a car is lower than the dimension of its configuration space, every configuration can be reached by a suitable sequence of controls, given that there are no obstacles.

We can relate this example to the previously defined Lie bracket operator (see Eq. (3.58)). As previously established, the Lie bracked allows us to calculate the vector field that describes the displacement achievable by alternately following two provided vector fields. Applied to the generators $\hat{\mathfrak{g}}_1$, $\hat{\mathfrak{g}}_2$ and $\hat{\mathfrak{g}}_3$ from the Lie algebra of the group of rigid transformations in plane,

[5] Ackerman steering configuration, see [Siegwart and Nourbakhsh, 2004, pp. 37ff.]



(a) Visualization of $[\hat{\mathfrak{g}}_1, \hat{\mathfrak{g}}_2]$ as a sequence of four controls that commute. Thus the start and the end configuration coincide. Example given for a holonomic robot.

(b) Visualization of $[\hat{\mathfrak{g}}_1, \hat{\mathfrak{g}}_3]$ as a sequence of four controls that do not commute. Executing these controls results in a sideways net displacement. Example given for a nonholonomic robot using a slip/skid steering configuration.

Figure 3.10: Visualization of the Lie bracket operation for commuting and non-commuting controls that correspond to the generators of the special Euclidean group $SE(2)$. In both cases translation and rotation are expressed with respect to the agent's body fixed reference frame where forward motion corresponds to movement into the positive x axis. Figures based on [Choset et al., 2005, p. 411, Figure 12.8].

$SE(2)$ (see Appendix A.2.1) it holds that

$$[\hat{\mathfrak{g}}_1, \hat{\mathfrak{g}}_2] = 0_{3\times3} \tag{3.76}$$

$$[\hat{\mathfrak{g}}_1, \hat{\mathfrak{g}}_3] = -\hat{\mathfrak{g}}_2. \tag{3.77}$$

Equation (3.76) shows that translation in x- and y-direction commute. However Eq. (3.77) shows that (a) x-translation and rotation do not commute but that (b) alternately executing said motions results in a y-translation which is exactly what we have established in our parallel parking example [Choset et al., 2005, Chapter 12.1.3]. A visualization for both Lie brackets is given in Fig. 3.10, based on the example calculations given in [Choset et al., 2005, pp. 409–411]. Figure 3.10a shows a holonomic robot executing commutative controls while Fig. 3.10b shows a nonholonomic robot executing controls that do not commute. The system depicted in Figure 3.10b uses a slip/skid steering configuration [Siegwart and Nourbakhsh, 2004, Chapter 2.3.2.3]. It can move its wheels individually, which makes driving straight as well as curved paths, and turning on the spot possible. However, like for the car, moving directly sideways is impossible for this robot.

3.8 Summary

In this chapter we have established the mathematical foundations for representing the configuration space of a mobile entity using matrix Lie groups. By representing the configuration space as such a Lie group, a smooth manifold with group structure, we can relate changes on the configuration space to the tangent space of said structure and ultimately represent motor controls

as tangent vectors. Due to the connection between the tangent space of a manifold and vector fields, we can relate motor control execution to following certain vector fields, i.e., interpret motor control execution to paths on the manifold. Relating motor controls to each other with respect to the Lie bracket, an operator that assesses the degree of commutativity, allows us to calculate the effect of non-commuting motor controls in a formal way.

4
Sensorimotor Agents

In this chapter, we build upon the concepts introduced in Chapter 3 and relate them to the definition of rational agents and their interaction with the environment with the aim to provide a thorough formal definition of a sensorimotor system. The structure is as follows: We start this chapter by establishing a set of assumptions in Section 4.2 regarding the environment, the agent, and the interaction between both that influences the further development. In Section 4.3 we propose an extrinsic view on the agent and its environment, allowing us to formally define the relationship between motor controls, sensor measurements and the agent's configuration space. In Section 4.4, we shift to an intrinsic view, discussing how the established concepts are utilized with respect to the agent control loop, which we adapt accordingly. The chapter concludes with a summary and an outlook to the next chapter in Section 4.5.

4.1 Synopsis

We initially establish a set of assumptions to be made with respect to the scenario, the agent, and the environment. We consider scenarios where a mobile agent moves through n-dimensional Euclidean space, which allows us to represent the corresponding configuration space using matrix Lie groups. We account for both holonomic and nonholonomic cases (see Section 3.7), where we assume the set of controls to be symmetric and a mechanism to be in place that returns feedback on their successful or partially successful execution. The relation between motor controls and sensor measurements is assumed to be locally smooth as well as consistent with respect to the associated configuration. We consider the interaction of a single agent with the static, continuous, and partially observable environment to be sequential, where, with respect to motor controls and sensor measurements, we cover deterministic scenarios as well as those in which either or both properties are subject to noise.

We then provide two distinct views on the scenario. The first one is an extrinsic view that considers the agent, the environment, and the interaction between those as viewed from an external observer, modeling them as an integrated and interdependent system. The second is an intrinsic view, where we restrict the knowledge to only those properties immediately observable by the agent or to those provided to the agent as a priori knowledge. While the extrinsic view allows us to establish a formal view on the functional relationship between all properties, the intrinsic view models the data available to the agent with respect to the task of bootstrapping.

For the extrinsic view, we establish a formal definition of the sensorimotor system where we model motor controls as vectors from a real vector space as well as an agent-specific subset of valid controls. We account for both holonomic and nonholonomic cases by defining a function that maps the potentially lower dimensional control space to a vector space with the same dimensionality as the configuration space. We establish a relation between these two spaces by defining a transition function that allows us to represent motor controls with respect to the Lie algebra of the configuration space. Parameterizing these functions allows us to model different types of motion behavior, i.e., different types of agents. Thus, executing a motor control corresponds to applying the group operation on the current configuration, where the closure of the group operation ensures that the agent can not leave the manifold. To model sensory feedback, we establish a functional relation between the configuration space and the set of all perceivable samples. Both the transition and the sample function are defined to allow the formal representation of noise, where in the former, noise impairs executed controls, i.e., the effect of motor controls does not match the assumed effect, and in the latter, noise impairs the perceived samples.

We then change the viewpoint to an intrinsic one, i.e., we shift the view from an overview on both the environment and the agent to an agent-centric one. As the agent only interacts with the environment through invocation of its sensors and actuators, the corresponding elements in the agent control loop have to be modeled accordingly. We modify the agent control loop introduced in Chapter 1, which will serve as a starting point for establishing the sensorimotor map in the next chapter. We close the chapter with a summary and an outlook in Chapter 5.

4.2 Preliminary Definitions and Assumptions

Within the scope of this thesis, we work with a certain set of assumptions with respect to the properties of the entity, the environment, and the interaction between both. We use the terms 'entity', 'system', and 'agent' synonymously, where all of these denote a coherent structure endowed with the means of perception and locomotion. The agent is situated in an environment that corresponds to n-dimensional Euclidean space. Thus, a distinct configuration of the agent with respect to its environment consists of its pose, i.e., its position and orientation. This allows us to represent the configuration space using matrix Lie groups (see Section 3.5), where motor controls, i.e., actions that change the agent's configuration, are associated with paths on the manifold.

Within the scope of this thesis, we consider both holonomic and nonholonomic cases (see Section 3.7). Thus, the number of controllable degrees of freedom will either match the number

of degrees of freedom of the configuration space (holonomic case) or will be lower (nonholonomic case).

With respect to the executable controls, we assume the zero control to be executable, i.e., the agent is able to execute a motor control that has no effect. In addition, we assume the set of executable motor controls to be symmetric, i.e., for any executable control its inverse exists that can also be executed and has the inverse effect [Choset et al., 2005, Chapter 12.2].

With respect to the application of controls, we assume a mechanism being in place that returns feedback on the degree of their successful execution. Within biological systems, this corresponds to proprioceptive feedback [Brodal, 2004, Chapter 6, pp. 145ff.], in robotic systems, this corresponds to odometers that are, as given by [Thrun et al., 2005, p. 23], *"sensors that measure the revolution of a robot's wheels. As such they convey information about the change of state"*.

To be able to infer the structure of the configuration space solely based on the relation between motor controls and sensor measurements, naturally an integral requirement is for such a structural relation between both to exist in the first place. Thus, sensor measurements have to correspond to certain properties, innate to the configuration they were perceived at, where the relation between a measurement and its corresponding configuration has to be consistent: two measurements perceived at the same configuration should be similar, where the degree of similarity naturally depends on whether we consider noisy or non-noisy sensors. However, measuring a constant signal or random noise would not be of much use. This is in line with [Kuipers and Levitt, 1988], where measurements are required to be *"distinctive enough to allow assimilation of the environmental structure. Open ocean and the neophytes view of desert or forest might fail to satisfy this requirement"*.

As established in the examples given in Section 1.3, the deliberative capabilities and thus the map representation have to be appropriately chosen to fit the task the rational agent is confronted with. We specify the properties of the configuration space using the properties of task environments proposed in [Russell and Norvig, 2003, Chapter 2.3].

We consider the scenario to be 'sequential', as the pose of the agent at a given point in time depends on previous poses and motor controls.

The configuration space is considered 'static' as its properties do not change over time. While the configuration of the agent (i.e., its pose) is subject to change, motor controls executed by the agent are the sole cause of theses changes. Thus, no external forces act upon the agent, which corresponds to the notion of a drift-free system (see Section 2.3.3 and [Choset et al., 2005, Chapter 12.2, p. 415] as well as [Arieşanu, 2015] for an example of a drift-free control system). A counterexample is a ship that is exposed to winds and currents, resulting in the net change of its configuration being a combination of the controls applied by the entity itself and the external forces. These assumptions are in line with the ones made in other bootstrapping scenarios, where [Pierce and Kuipers, 1997] consider the system in question to be situated in a *"static environment"* where *"nothing changes when the control signals are all zero"*.

As pointed out in [Russell and Norvig, 2003, p. 42], the attributes 'discrete' and 'continuous' have to be considered separately with respect to the configuration space, motor controls, sensor measurements, and how passing of time is modeled. We consider the former three to be 'continuous', where for the configuration space this is implied by the smoothness of the manifold

representation. Motor controls are modeled with respect to vector fields that facilitate smooth state transitions. We assume sensor measurements to be continuous, and we assume a locally smooth relation between the configuration space and the sensor measurements. Thus, we assume that there exist local patches where sensor measurements vary smoothly across the configuration space and do not exhibit discontinuities. Ultimately, this translates to a correspondingly smooth relation between motor controls and sensor measurements, i.e., the assumption that small motor controls result in small changes in the perceivable sensor inputs. This requirement has also been identified as integral for the learning task by [Pierce and Kuipers, 1997] as *"learning methods rely on generic properties of the robots world such as almost-everywhere smooth effects of motor control signals on sensory features"*, which is explained as the *"derivatives with respect to time of the sensor values can be approximated by linear functions of the motor control vector"*. This is in line with the assumptions made by [Philipona et al., 2003] as they assume that *"the environment imposes a smooth enough (meaning we will consider the problem only in a region excluding the singularities of ψ) relation between sensory signals S and motor commands M"*, where in this case ψ denotes the functional relationship between the environment, motor commands, and sensory signals. This is closely related to the assumption made by [Olsson et al., 2005], where the system has *"no innate knowledge regarding the modality or representation of the sensory input and the actuators"* and thus *"relies on generic properties of the robots world such as piecewise smooth effects of movement on sensory changes"*. In [Olsson et al., 2006], this property is elaborated on with respect to light sensors, where *"the world is made up of continuous objects over which brightness varies smoothly"*. The property of being locally smooth is especially important when utilizing sensors that provide discontinuous feedback. As an example, we consider the measurements provided by a range sensor that will 'jump' when encountering an edge, creating a discontinuity in the, otherwise smooth, structure of the relation between the environment and the measurement, i.e., a discontinuity in the corresponding functional relation [Thrun et al., 2005, Chapter 6.3.5]. However, as long as the aforementioned locally smooth patches exist, we can model sensor measurements, i.e., the sensor's behavior in these regions.

We consider the passing of time to be modeled in a 'discrete' fashion. Perceiving the environment and deciding on the next control is instantaneous and only executing a motor control advances the world from one discrete time step to the next. With respect to the availability of information, the configuration space is 'partially observable'. Thus, the agent can only perceive sensor measurements associated to its current configuration. Regarding the question whether the configuration space is 'deterministic' or 'stochastic', we account for both cases. In the former, sensory inputs are non-noisy and the effects motor controls have on the agent's pose are deterministic. In the latter, either sensor measurements, motor controls, or both are subject to noise. The configuration space will not account for any other agents, i.e., we only consider 'single agent' scenarios. Thus, the agent will not interact with any other agents, neither in a cooperative, nor an adversarial way.[1]

[1] It has to be noted that within the simulation environment (see Section 7.4), all agents active in a distinct experiment/iteration inhabit the same 'world'. They are, however, not able to interact with each other by means of communication or collide with each other, effectively making this a 'single agent' scenario.

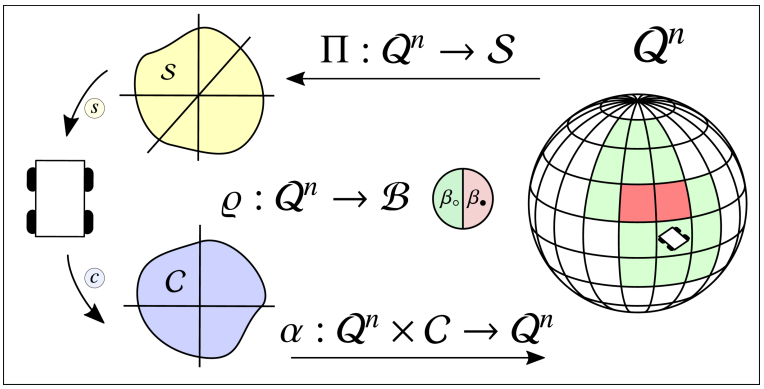


Figure 4.1: Extrinsic view of the sensorimotor system. The configuration space $\mathcal{Q}^n$ is composed of unobstructed and obstructed configurations (not shown) as defined by the agent specific occupancy function ϱ that maps elements from $\mathcal{Q}^n$ into the boundary set $\mathcal{B}$. The sample function Π allows the agent to perceive sensor measurements by mapping configurations from $\mathcal{Q}^n$ to samples from $\mathcal{S}$. The transition function α allows the agent to change its configuration by applying valid controls $c \in C_\odot$ to the current configuration.

In the next section, we define the extrinsic view of the agent, i.e., the view an external observer has of the agent and the environment it is situated in.

4.3 Extrinsic View

In this section, we propose an extrinsic view of the sensorimotor system, i.e., a view that accounts for both the agent and the environment and allows us to formally model the interaction between both. The term 'agent' in this context refers to the concept of a rational agent introduced in Section 1.1 and the corresponding agent loop as given in Algorithm 1. Within the context of this thesis, we formally define a sensorimotor system as the 6-tuple

$$\mathrm{S}_{(\Leftrightarrow)} = (\mathcal{Q}^n, \varrho, C_\odot, \mathcal{S}, \alpha, \Pi). \tag{4.1}$$

As previously established in Section 3.2, we denote $\mathcal{Q}^n$ as the configuration space that corresponds to an arbitrary but fixed matrix Lie group (see Section 3.5). The configuration space is the union of both unobstructed and obstructed configurations (see Eqs. (3.12) and (3.13)), where both subsets are defined with respect to the entity-specific occupancy function ϱ (see Eq. (3.10)). The set of all valid controls with respect to the agent's actuator is denoted by $C_\odot$. The sample set $\mathcal{S}$ denotes the set of all sensory inputs the agent can perceive. The relation between the configuration space $\mathcal{Q}^n$ and the set of valid controls $C_\odot$ is specified by the actuator- and thus agent-specific transition function α, which encodes the effect of motor controls on configurations. The relation between the configuration space $\mathcal{Q}^n$ and the set of all perceivable samples is specified by the sensor- and thus agent-specific sample function Π, which encodes the relation between configurations and perceived samples, i.e., it allows to associate samples to distinct elements of the configuration space. The interaction between these spaces and functions is visualized in Fig. 4.1. In the following, we give a more in-depth definition of these components where we aim for modeling the interaction with the environment as abstractly as possible. Thus both the

channel that allows the agent to act in the environment (given by the transition function) and the channel that allows the entity to perceive information about its current state (given by the sample function) will be modeled as m- and k-dimensional real vectors, respectively. Conceptually, the sensorimotor system can be related to the notion of sensorimotor contingencies (SMCs) given in [ORegan and No, 2001], where SMCs with respect to visual perception are proposed as the *"structure of the rules governing the sensory changes produced by various motor actions"*. This corresponds to the definition of the sensorimotor environment introduced by [Buhrmann et al., 2013], which *"constitutes the set of all possible sensory dependencies on motor states (s, m) for a particular type of agent and environment."*. These properties are accordingly represented by defining the configuration space $\mathcal{Q}^n$, the set of valid controls $C_{\odot}$, the set of sensory inputs $\mathcal{S}$, and their relation given by the functions α and Π.

4.3.1 Configuration Space

As established in Chapter 3, within the scope of this thesis we model sensorimotor systems whose configuration space (see Section 3.2) can be represented as a smooth manifold (see Section 3.3). As we explicitly focus on scenarios where invoking motor controls corresponds to changes in position and orientation, we represent the configuration space as a matrix Lie group, as these Lie groups allow for the modeling of translations and rotations in n-dimensional Euclidean space as established in Sections 3.5 and 3.6. We denote $\mathcal{Q}^n$ the configuration space that is given as an arbitrary but fixed matrix Lie group. Portions of $\mathcal{Q}^n$ being inaccessible due to physical constraints are accounted for by the definition of the occupancy function (Eq. (3.10)) and the corresponding obstructed and unobstructed subsets of $\mathcal{Q}^n$ (see Eqs. (3.12) to (3.13)).

It has to be noted that the configuration space may represent a portion of a larger environment, where an example is the mobile robot depicted in Section 1.3. Its configuration space corresponds to $SE(2)$, the Lie group representing translation and rotation in 2-dimensional Euclidean space. However, $SE(2)$ is isomorphic to a subset of $SE(3)$, i.e., the matrix Lie group representing translation and rotation in 3-dimensional Euclidean space. Thus, the manifold the entity can traverse is embedded within another structure. While this is obvious to an external observer, the agent has no means of either detecting this relationship or accessing portions of the ambient space not contained within its configuration space.

The property of $\mathcal{Q}^n$ being composed of empty and occupied portions has direct practical implications with respect to the sensorimotor map, as from a modeling standpoint physical constraints have to be accounted for.

The case where $\mathcal{Q}^n$ represents only a subset of a larger environment, however, is more of a philosophical consideration: Obviously the internal representation the entity needs in order to navigate through its environment has only to account for configurations it could actually reach, which per definition corresponds to its configuration space. This means that while we, as external observers, in this case could model $\mathcal{Q}^n$ utilizing $SE(3)$, it would be pointless as everything other than the part corresponding to $SE(2)$ could not be utilized by the entity anyway.

For the remainder of this chapter, we assume that the matrix Lie group corresponding to the configuration space is known.

4.3.2 Control Space

As established in Section 1.1, a rational agent is equipped with a set of actuators $\mathcal{A}$, which are components that allow the agent to move in its environment. We model actuators as abstract control interfaces that relate control vectors to motor actions, and thus ultimately to vector fields on $\mathcal{Q}^n$. This abstract definition allows us to model arbitrary systems using only one actuator, as the actuator directly encodes the effect motor controls have with respect to the agents's pose. Thus, the subtleties of whether a certain change in position and orientation is caused by utilizing one motor or two motors, are hidden within the implementation. In the remainder of this thesis the agent is equipped with a single actuator a. Thus and it holds that

$$\mathcal{A} = \{\, a \,\}. \tag{4.2}$$

Naturally, invoking the actuator a with an control vector $c \in \mathbb{R}^m$, changes the agent's position and orientation. As established in Sections 3.6 and 3.7, the relation between actuator commands as vectors $c \in \mathbb{R}^m$ and configuration changes is encapsulated by the matrix Lie group specific interpretation of the $\boxplus$- and $\boxminus$-operators (Eqs. (3.66) and (3.67)) as well as the mapping from $\mathbb{R}^m$ to the Lie algebra, given by the transformation τ (Eqs. (3.70) and (3.71)). However, we have yet to provide a more formal definition of the set of valid controls as well as of the mapping into the Lie algebra. To generalize the notation we substitute $\mathbb{R}^m$ with C, thus

$$C \equiv \mathbb{R}^m, \tag{4.3}$$

where we denote C the control space of the entity, where m corresponds to the arbitrary but fixed DoF of the actuator. As established in Section 4.2, we consider both holonomic and nonholonomic systems (see Section 3.7) but exclude overactuated systems, i.e., systems for which the DoF of the actuator is greater than the DoF of the configuration space. In the holonomic case, the number of controllable degrees of freedom matches the number of degrees of freedom of the configuration space, which corresponds to the case where $\dim(C) = \dim(\mathcal{Q}^n)$. In the nonholonomic case, the number of controllable degrees of freedom is lower than the number of degrees of freedom of the configuration space, which corresponds to the case where $\dim(C) < \dim(\mathcal{Q}^n)$. To account for both cases we denote the extended control space C_* as the Euclidean space $\mathbb{R}^n$ whose dimensions match those of the configuration space

$$C_* \equiv \mathbb{R}^n \quad \text{with} \quad n = \dim(\mathcal{Q}^n). \tag{4.4}$$

In both cases, the mapping τ_* from C to C_* is given as

$$\tau_* : C \to C_* \tag{4.5}$$

with

$$\tau_*(c) = \begin{bmatrix} I_{m\times m} \\ 0_{(n-m)\times m} \end{bmatrix} c = T_* c = c_* \quad \text{with} \quad c \in C,\ c_* \in C_*. \tag{4.6}$$

Here, m and n denote the dimensions of C and C_*, respectively. The transformation matrix associated to τ_* is given by T_*, where $I_{m\times m}$ corresponds to the $m\times m$ identity matrix and $0_{(n-m)\times m}$ to the $(m-n)\times n$ zero matrix. The function τ_* thus maps vectors

$$c = [\, c_1, \cdots, c_m \,]^T \in (C \equiv \mathbb{R}^m)$$

to vectors

$$c_* = [\, c_1, \cdots, c_m, 0_1, \cdots, 0_{n-m} \,]^T \in (C_* \equiv \mathbb{R}^n).$$

If $m = n$ it holds that $\mathbb{R}^m = \mathbb{R}^n$, accordingly $C = C_*$ and thus T_* corresponds to the identity matrix $I_{m\times m}$. If $m < n$, it holds that C corresponds to only a subset of C_*.

Though we defined the control space C to correspond to $\mathbb{R}^m$, we have to account for actuator-specific constraints that may limit valid controls to a subset of C. To allow for a generic modeling of these constraints, we define the entity- and actuator-specific function

$$\text{valid} : C \to \{\, true,\ false \,\} \tag{4.7}$$

that allows assessing whether a given control vector $c \in C$ is usable by the actuator. Notably, this only states that the control vector can be processed by the actuator. It does not indicate that the control can be executed without the agent encountering an obstacle. We denote the set of valid controls as the subset $C_\odot \subseteq C$ given as

$$C_\odot = \{\, c \in C \,|\, \text{valid}(c) = true \,\}. \tag{4.8}$$

As established in Section 4.2, we require the zero control to be included in C as well as C to be symmetric. Thus, it holds that $0_\odot \in C_\odot$ as well as $\forall c \in C_\odot : -c \in C_\odot$. However, the inverse of a control being valid does not necessarily mean that it is executable: even if the agent can execute the control $c \in C_\odot$ at a given configuration without encountering an obstacle, the same does not necessarily hold for its inverse $-c \in C_\odot$. Applying Eq. (4.5) to this set yields

$$C_\circledast = \{\, c_* \in C_* \,|\, c_* = \tau_*(c),\ c \in C_\odot \,\} \tag{4.9}$$

as the image of $C_\odot$ under τ_*. As neither $C_\odot$ nor $C_\circledast$ are guaranteed to have a regular shape, we define an envelope set on C_*, which guarantees that the image of all valid controls $c \in C_\odot$ under τ_* is contained within it. We denote this envelope $C_\circledcirc \subseteq C_*$ and define it as the Euclidean n-ball

$$C_\circledcirc = \{\, c_* \in C_* \,|\, \|c_*\| \leq \epsilon_\circledcirc \,\} \quad \text{where} \quad \forall c \in C_\circledast : \epsilon_\circledcirc \geq \|c\|. \tag{4.10}$$

Here, the radius $\epsilon_\circledcirc \in \mathbb{R}^+$ is chosen to be greater or equal to the magnitude of every control in $C_\circledast$. This guarantees that $C_\circledcirc$ contains all controls in $C_\circledast$ and, thus, it holds that $C_\circledast \subseteq C_\circledcirc$. A visualization of the relations established so far is given in Figure 4.2. We utilize the envelope $C_\circledcirc$ as a fundamental building block when constructing the sensorimotor map in Chapter 5.

4.3.3 Transition Function

The transition function models the relation between q_i and q_j with $q_i, q_j \in Q^n$ as the effect of executing a control $c_\odot \in C_\odot$ (see Eq. (4.8)). We denote this transition function as

$$\alpha : Q^n \times C_\odot \to Q^n. \tag{4.11}$$

As established in Chapter 3 and Section 4.3.1, we represent the configuration space Q^n as a matrix Lie group. This allows us to model the relation between configuration space elements

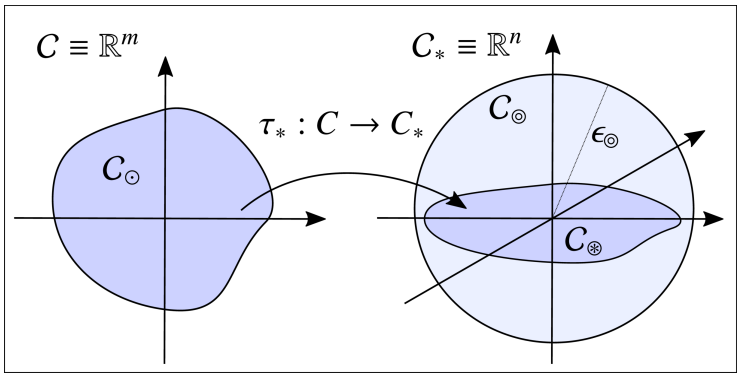


Figure 4.2: Relationship between the control space C, the set of valid controls $C_\odot$, and the extended control space C_*. Here, $C_\circledast$ denotes the image of $C_\odot$ under τ_*. The envelope $C_\circledcirc$ in C_* is given as the Euclidean n-ball with radius $\epsilon_\circledcirc$ that contains $C_\circledast$.

$q \in Q^n$ and elements of its Lie algebra, i.e., its tangent space $T_{id}Q^n$ using the specific $\boxplus_{[\text{Lie}]}$- and $\boxminus_{[\text{Lie}]}$-operators for matrix Lie groups given in Eqs. (3.66) and (3.67), respectively. These operators relate tangent space vectors to paths on the manifold. Combining $\boxplus_{[\text{Lie}]}$ and $\boxminus_{[\text{Lie}]}$ with the established function τ_* (see Eqs. (3.70) and (3.71)) allows us to relate elements from C to elements from Q^n. In the following, we cover the mapping from C to Q^n as well the mapping from Q^n to C and $C_\odot$.

4.3.3.1 From Controls to Group Elements

We initially define the forward mapping, i.e., the mapping from controls $c \in C_\odot$ to group elements $q \in Q^n$. As an intermediate step, we look at the mapping from the extended control space C_* to the tangent space $T_{id}Q^n$. In this case both C_* and $T_{id}Q^n$ are n-dimensional vector spaces over $\mathbb{R}$. Thus, elements from C_* and $T_{id}Q^n$ can be represented as linear combinations of their respective basis vectors. For C_* these are the standard basis vectors $\hat{e}_1, \cdots, \hat{e}_n$ for the Euclidean space $\mathbb{R}^n$, while for $T_{id}Q^n$ these basis vectors correspond to the Lie group specific generators $\hat{\mathfrak{g}}_1, \cdots, \hat{\mathfrak{g}}_n$ (see Section 3.5). Thus, we define the mapping from C_* to $T_{id}Q^n$ as

$$\tau_\cup : C_* \to T_{id}Q^n. \tag{4.12}$$

A linear map between two n-dimensional vector spaces can be represented as a matrix multiplication, thus

$$\tau_\cup(c_*) = T_\cup c_* = \mathfrak{q} \quad \text{with} \quad c_* \in C_* \text{ and } \mathfrak{q} \in T_{id}Q^n, \tag{4.13}$$

where $T_\cup$ denotes a real $n \times n$ matrix and $\mathfrak{q}$ corresponds to the short notation for a Lie algebra element associated to c_* according to Eq. (3.57). In the following, we require $T_\cup$ to have full rank, i.e., it has to hold that the rank of $T_\cup$ equals n. To generate the corresponding group element, we define

$$\tau_\triangleright : C_* \to Q^n \tag{4.14}$$

as a composition of the exponential map (Eq. (3.62)) and $\tau_{\circlearrowleft}$ (Eq. (4.12)) with

$$\tau_{\triangleright} = \exp \circ \tau_{\circlearrowleft} \tag{4.15}$$

which for a given $c_* \in C_*$ yields

$$\begin{aligned} \tau_{\triangleright}(c_*) &= (\exp \circ \tau_{\circlearrowleft})(c_*) \\ &= \exp(T_{\circlearrowleft} c_*) \\ &= \exp(\mathfrak{q}) \\ &= q \end{aligned} \tag{4.16}$$

where $\mathfrak{q} \in T_{id}Q^n$ and $q \in Q^n$. To construct the mapping from the control space C to Q^n, we compose $\tau_{\triangleright}$ with the previously defined τ_* given by Eqs. (4.5) and (4.6). Thus, we define

$$\tau_{\blacktriangleright} : C \to Q^n \tag{4.17}$$

as the composition

$$\tau_{\blacktriangleright} = \tau_{\triangleright} \circ \tau_* \quad \text{i.e.} \quad \tau_{\blacktriangleright} = \exp \circ \tau_{\circlearrowleft} \circ \tau_* \tag{4.18}$$

which yields

$$\begin{aligned} \tau_{\blacktriangleright}(c) &= (\tau_{\triangleright} \circ \tau_*)(c) \\ &= (\exp \circ \tau_{\circlearrowleft} \circ \tau_*)(c) \\ &= \exp(T_{\circlearrowleft} T_* c) \\ &= \exp(T_{\circlearrowleft} c_*) \\ &= \exp(\mathfrak{q}) \\ &= q, \end{aligned} \tag{4.19}$$

where $c \in C$, $c_* \in C_*$, $\mathfrak{q} \in T_{id}Q^n$, and $q \in Q^n$. This transformation accounts for both the holonomic and the nonholonomic case as τ_*, i.e., the dimension of the transformation matrix T_* can be adjusted accordingly (see Eq. (4.6)). To model the effect of noisy controls, we introduce a corresponding m-dimensional noise vector denoted by ϵ_c, which is sampled from a zero-mean normal distribution with actuator-specific covariance matrix $\Sigma^c_{m\times m}$, thus

$$\epsilon_c \sim \mathcal{N}(\, 0_m,\, \Sigma^c_{m\times m}\,) \tag{4.20}$$

where 0_m denotes the m-dimensional zero vector. The potentially noisy equivalent $\tilde{c}$ for a given control $c \in C_{\odot}$ is then given by

$$\tilde{c} = \; c + \epsilon_c \sqrt{||c||} \tag{4.21}$$

where in the non-noisy case, i.e., when $\Sigma^c_{m\times m}$ is set to zero, naturally $\tilde{c} = c$. The noise is applied additively, but is scaled to prevent the entity from moving when no control has been applied, i.e., when applying the zero control $0_{\odot}$. While c is chosen as as element from $C_{\odot}$, it is not guaranteed

that the same holds for $\tilde{c}$, as adding the noise term may result in a control that is not within $C_\odot$ but only in C. This has to be considered when defining functions working with $\tilde{c}$. It has to be noted that we apply noise to the control c instead to the tangent space element $(\tau_\cup \circ \tau_*)(c) = \mathfrak{q}$. This limits the application of noise to controllable degrees of freedom dimensions. This has been done to prevent the agent from moving out of its native configuration space and thus prevent the agent from reaching configurations it could never naturally, i.e., using its means of locomotion, return from. This is closely related to the assumption of not modeling drift vector fields (see Section 4.2), as applying noise to elements from C_* would correspond to modeling a fluctuating drift-vector field that introduces potential non-compensable disturbances. An example would be to apply 'sideways' noise to a robot that is only able to move forward and backward. Using its means of locomotion, the robot could never compensate the introduced disturbance and could never return to its starting configuration.

With both the control space and the individual transformations from and to the configuration space being defined, we can now explicitly model the transition function as

$$\begin{aligned}\alpha(q_i, c) &= q_i \bullet \tau_\blacktriangleright(c + \epsilon_c \sqrt{\|c\|}) \\ &= q_i \bullet \tau_\blacktriangleright(\tilde{c}) \\ &= q_i \bullet \exp(T_\cup T_* \tilde{c}) \\ &= q_i \bullet \exp(\tilde{\mathfrak{q}}) \\ &= q_i \bullet \tilde{q} \\ &= \tilde{q}_j\end{aligned} \tag{4.22}$$

with $c \in C_\odot$, $\tilde{c} \in C$ and $q_i, \tilde{q}_j, \tilde{q} \in \mathcal{Q}^n$. In the noisy case $\tilde{q}_j$ naturally differs from the configuration the agent intended to move, given by $q_j = q_i \bullet \tau_\blacktriangleright(c)$. Figure 4.3 visualizes the effect of the transition function with respect to a path on $\mathcal{Q}^n$ and its individual components.

However, we have to account for cases where the movement of an entity is obstructed (see Eq. (3.10)). This means that for a given $q_i \in \mathcal{Q}^n$ there might exist a $c \in C_\odot$ that is not fully executable, i.e., the corresponding control $\tilde{c} \in C$, and thus the transformation $\tilde{q}$ can only be executed partially or, in an extreme case, can not be executed at all. To represent these cases, we relate the transition function α to paths (see Eq. (3.17)) on the manifold $\mathcal{Q}^n$ by defining

$$\text{path} : \mathcal{Q}^n \times C_\odot \to \Gamma_{\mathcal{Q}^n}, \tag{4.23}$$

where $\Gamma_{\mathcal{Q}^n}$ denotes the set of all paths on $\mathcal{Q}^n$ as defined by Eq. (3.19). Applying Eq. (4.23) to a given $q_i \in \mathcal{Q}^n$ and $c \in C_\odot$ yields

$$\text{path}(q_i, c) = \vec{\gamma}_{(q_i, \tilde{q}_j)} \quad \text{with} \quad \vec{\gamma}_{(q_i, \tilde{q}_j)} : [0, 1] \to \mathcal{Q}^n, \tag{4.24}$$

where it naturally holds that $\vec{\gamma}_{(q_i, \tilde{q}_j)}$ is the locally shortest path, i.e., a geodesic on $\mathcal{Q}^n$ (see Section 3.4), which can be expressed in terms of the transition function in Eq. (4.22) with

$$\begin{aligned}\vec{\gamma}_{(q_i, \tilde{q}_j)}(t) &= \alpha(q_i, tc) \quad \text{with} \quad t \in [0, 1] \\ &= q_i \bullet \exp(T_\cup T_* t\tilde{c}),\end{aligned} \tag{4.25}$$

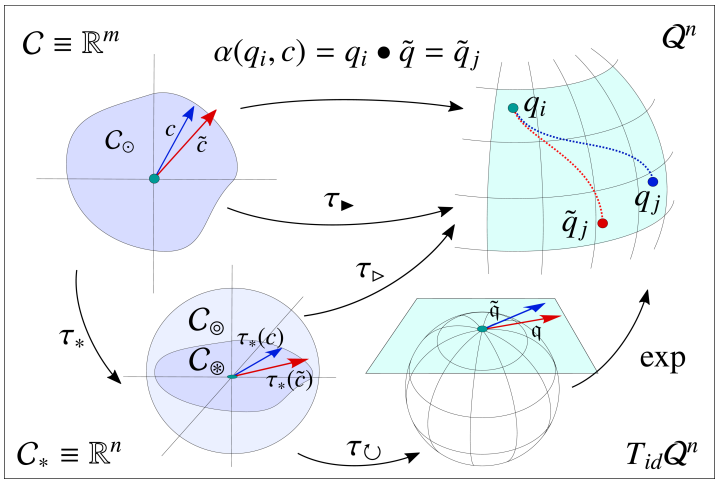


Figure 4.3: Transition function, modeled as a composition of τ_*, $\tau_\cup$, the exponential map and the group operation. Depiction of the effect of noisy (red) and non-noisy (blue) controls. In the noisy case, noise is applied to a control c from the set of valid controls $C_\odot$. The noisy control $\tilde{c}$ is then mapped into the extended control space C_* by utilizing τ_*. Utilizing $\tau_\cup$ maps $\tau_*(\tilde{c})$ into $T_{id}Q^n$, the tangent space of Q^n. The tangent vector $\tilde{q}$ is transformed into the group element $\tilde{q}$ by utilizing the exponential map. Applying the group operation to the current configuration q_i and the transformation $\tilde{q}$ yields the new configuration $\tilde{q}_j$, where the path between q_i and q_j is a geodesic on Q^n. As a comparison, q_j is depicted, the configuration reached when applying the non noisy control c, i.e., the corresponding group element $\exp(\mathfrak{q})$ to q_i.

where naturally it holds that

$$\vec{\gamma}_{(q_i,\tilde{q}_j)}(0) = \alpha(q_i, 0c) = \alpha(q_i, 0_\odot) = q_i, \tag{4.26}$$
$$\vec{\gamma}_{(q_i,\tilde{q}_j)}(1) = \alpha(q_i, 1c) = \alpha(q_i, c) = \tilde{q}_j. \tag{4.27}$$

We are now interested in the portion of $\vec{\gamma}_{(q_i,\tilde{q}_j)}$ that is unobstructed. Thus, we apply the previously established function $\text{free}_\otimes$ (see Eq. (3.24)) to Eq. (4.23) which yields

$$\text{free}_\otimes \circ \text{path} : Q^n \times C_\odot \to [0, 1]. \tag{4.28}$$

According to Eq. (3.25), this yields

$$(\text{free}_\otimes \circ \text{path})(q_i, c) = \begin{cases} 0, & \text{if} \quad q_i \in Q^n_\bullet \\ 1, & \text{if} \quad \text{free}_\circ(\vec{\gamma}_{(q_i,\tilde{q}_j)}) = \mathit{true} \\ t^\otimes \in (0, 1), & \text{otherwise}, \end{cases} \tag{4.29}$$

where $t^\otimes$ is accordingly to Eq. (3.26) given as

$$\forall t \in [0, t^\otimes] : \vec{\gamma}_{(q_i,\tilde{q}_j)}(t) \in Q^n_\circ \quad \text{and} \quad \vec{\gamma}_{(q_i,\tilde{q}_j)}(t^\otimes + \epsilon) \in Q^n_\bullet. \tag{4.30}$$

We define the partial transition function $\alpha_\otimes$ that executes a given control $c \in C_\odot$ either until full execution or encountering an obstruction as

$$\alpha_\otimes : Q^n \times C_\odot \to Q^n \tag{4.31}$$

that, with respect to Eqs. (4.11) and (4.28) to (4.30), is given as

$$\begin{aligned}\alpha_{\otimes}(q_i, c) &= \alpha(q_i, t^{\otimes} c) \\ &= q_i \bullet \tau_{\blacktriangleright}(t^{\otimes} \tilde{c}) \\ &= q_i \bullet \exp(T_{\circlearrowleft} T_* t^{\otimes} \tilde{c}) \\ &= q_i \bullet \tilde{q}^{\otimes} \\ &= \tilde{q}_j^{\otimes}.\end{aligned} \tag{4.32}$$

Here, $\tilde{q}^{\otimes}$ denotes the group element from $\mathcal{Q}^n$ that corresponds to the transformation associated to the control c that has been scaled with the appropriate length $t^{\otimes}$ of the maximal free path between q_i and $\tilde{q}_j$. Here $\tilde{q}_j^{\otimes}$ denotes the corresponding configuration in $\mathcal{Q}^n$ between q_i and $\tilde{q}_j$. To allow the entity to incorporate the information to which degree a given control has been executed, we define the partial transition function with proportional feedback as

$$\alpha_{\otimes}^{\triangleleft} : \mathcal{Q}^n \times C_{\odot} \to \mathcal{Q}^n \times [0, 1]. \tag{4.33}$$

With respect to Eqs. (4.28) and (4.31), this corresponds to

$$(\alpha_{\otimes} \times (\text{free}_{\otimes} \circ \text{path})) : \mathcal{Q}^n \times C_{\odot} \to \mathcal{Q}^n \times [0, 1], \tag{4.34}$$

which yields

$$\begin{aligned}\alpha_{\otimes}^{\triangleleft}(q_i, c) &= (\alpha_{\otimes} \times (\text{free}_{\otimes} \circ \text{path}))(q_i, c) \\ &= (\, \alpha_{\otimes}(q_i, c),\, (\text{free}_{\otimes} \circ \text{path})(q_i, c)\,) \\ &= (\, q_i \bullet \tilde{q}^{\otimes},\, t^{\otimes}\,) \\ &= (\, \tilde{q}_j^{\otimes},\, t^{\otimes}\,).\end{aligned} \tag{4.35}$$

Intuitively, this function expresses the effect of an entity gradually applying a control $c \in C_{\odot}$ until either an obstacle prevents the control from being applied any further (in which case $t^{\otimes} < 1$) or until the control has been applied completely (in which case $t^{\otimes} = 1$ holds). In both cases, corresponding feedback is returned. This feedback is independent of the nature of the trajectory as arbitrary shaped paths on $\mathcal{Q}^n$ are expressed as tangent vectors in $T_{id}\mathcal{Q}^n$, i.e., correspond to linear combinations of the generators of the Lie algebra (see Eq. (3.56)) and thus to geodesics on $\mathcal{Q}^n$. In addition, the feedback $t^{\otimes}$ is linear with respect to the geodesics on $\mathcal{Q}^n$, as the mapping $\tau_{\blacktriangleright}$ (see Eq. (4.17)) is composed of the linear transformations τ_* (see Eq. (4.5)), $\tau_{\circlearrowleft}$ (see Eq. (4.12)), and the matrix exponential Eq. (3.62). Thus, a feedback of $t^{\otimes} = 0.25$ would indicate that only 25% of the scheduled path has been executed. In case of noisy controls, it has to be considered that the feedback $t^{\otimes}$ is generated based on the noisy path and that utilizing $t^{\otimes}$ in conjunction with the non-noisy original control $c \in C_{\odot}$ to calculate the traveled path will introduce an error. This, however, is a desired effect considering the partial transition function with proportional feedback (Eq. (4.33)) as an abstract odometry model. According to [Thrun et al., 2005, p. 23], odometers *"are sensors that measure the revolution of a robot's wheels. As such they convey information about the change of state. Even though odometers are sensors, we will treat odometry as control data, since they measure the effect of a control action"*. Thus, even if the planned control

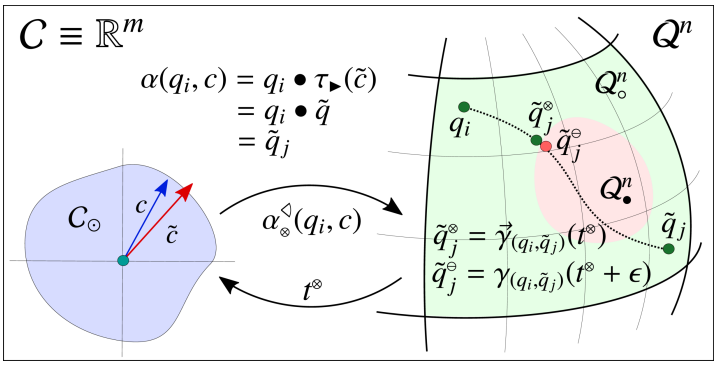


Figure 4.4: Partial transition function with feedback. It models the execution of a given control until either the control has been fully executed or an obstruction in Q^n has been encountered. It as well models the return of a corresponding proportional feedback. When utilizing $\alpha(q_i, c)$, the noisy control $\tilde{c}$ is mapped to $\tilde{q} \in Q^n$ by using $\tau_{\blacktriangleright}$. The end configuration $\tilde{q}_j$ is calculated by applying the $\tilde{q}$ to the starting configuration q_i, using the group operation. The path from q_i to $\tilde{q}_j$ is described by the geodesic $\vec{\gamma}_{(q_i,\tilde{q}_j)}$. The free portion of $\vec{\gamma}_{(q_i,\tilde{q}_j)}$ is calculated as $t^{\otimes} \in [0, 1]$ by the composition (free$_{\otimes}$ $\circ$ path). Utilizing $t^{\otimes}$ enables the calculation of $\tilde{q}^{\otimes}_j$ using $\alpha_{\otimes}(q_i, c)$. The proportional feedback returned corresponds to $t^{\otimes}$.

matches the returned feedback, the effect of the executed control might still be subject to noise. Figure 4.4 visualizes the partial transition function with proportional feedback with respect to the transformations on Q^n.

Another interesting property of the transition functions Eq. (4.11) is that the effect that controls from $C_\odot$ have on elements from a given matrix Lie group Q^n is entirely determined by the matrix $T_\circlearrowleft$. This allows us to define different types of mobile entities, i.e., different types of motor capabilities, by appropriately selecting the components in $T_\circlearrowleft$. An example that illustrates this relation is given in Appendix B.

4.3.3.2 From Group Elements to Controls

We now define the reverse mapping, i.e., the mapping between group elements $q \in Q^n$ and controls $c \in C_\odot$. As an intermediate step, we look at the mapping from the tangent space $T_{id}Q^n$ to the extended control space C_*. According to Eq. (4.12), the mapping from C_* to $T_{id}Q^n$ is given by the $n \times n$ matrix $T_\circlearrowleft$. As we required $T_\circlearrowleft$ to have full rank, we know that $T_\circlearrowleft$ is invertible. Thus, it holds that the mapping

$$\tau_\circlearrowleft : T_{id}Q^n \to C_* \tag{4.36}$$

is given as

$$\tau_\circlearrowleft(\mathfrak{q}) = T_\circlearrowleft \mathfrak{q} = T^{-1}_\circlearrowleft \mathfrak{q} = c \quad \text{with} \quad \mathfrak{q} \in T_{id}Q^n \text{ and } c \in C_*. \tag{4.37}$$

To apply this operation on group elements, we define

$$\tau_{\blacktriangleleft} : Q^n \to C_* \tag{4.38}$$

as a composition of $\tau_{\cup}$ (Eq. (4.36)) and the matrix logarithm (Eq. (3.63)) with

$$\tau_{\blacktriangleleft} = \tau_{\cup} \circ \log \tag{4.39}$$

which for a given $q \in \mathcal{Q}^n$ yields

$$\begin{aligned} \tau_{\blacktriangleleft}(q) &= (\tau_{\cup} \circ \log)(q) \\ &= T_{\cup} \log(q) \\ &= T_{\cup} \mathfrak{q} \\ &= c, \end{aligned} \tag{4.40}$$

where $\mathfrak{q} \in T_{id}\mathcal{Q}^n$ and $c \in C_*$. The mapping from C_* to C corresponds to the composition of $\tau_{\blacktriangleleft}$ and the inverse of the previously defined τ_* given by Eqs. (4.5) and (4.6) with

$$\tau_*^{-1} \circ \tau_{\blacktriangleleft} : \mathcal{Q}^n \to C. \tag{4.41}$$

In the holonomic case the dimensions of $C \equiv \mathbb{R}^m$ and $C_* \equiv \mathbb{R}^n$ coincide with $m = n$ (see Section 4.3.2) and thus T_* corresponds to the identity matrix $I_{m\times m}$. In this case it holds that

$$\begin{aligned} (\tau_*^{-1} \circ \tau_{\blacktriangleleft})(q) &= T_*^{-1} \tau_{\blacktriangleleft}(q) \\ &= I_{m\times m} \tau_{\blacktriangleleft}(q) \\ &= \tau_{\blacktriangleleft}(q). \end{aligned} \tag{4.42}$$

In the nonholonomic case, however, $m \neq n$ and thus τ_* is not invertible as T_* is not a square but (as given by Eq. (4.6)) an $n \times m$ matrix instead. A possibility, in this case, is to utilize the $m \times m$ portion of T_*, denoted as $T_{(m\times m)^*}$, and pad the matrix with zeros, which yields

$$\begin{aligned} (\tau_*^{-1} \circ \tau_{\blacktriangleleft})(q) &= \left[T^{-1}_{(m\times m)^*} \; 0_{m\times(n-m)} \right] \tau_{\blacktriangleleft}(q) \\ &= \left[I_{m\times m} \; 0_{m\times(n-m)} \right] \tau_{\blacktriangleleft}(q) \end{aligned}$$

However, this calculation is only valid if $c_* = \tau_{\blacktriangleleft}(q)$ with $c_* \in C_*$ is an element of $C_{\circledast}$, i.e., if the $n - m$ elements of $\tau_{\blacktriangleleft}(q)$ are zero. This corresponds to the case where an $c \in C$ exists for which $\tau_{\blacktriangleright}(c) = q$. If such a control does not exist, c_* is not an element of $C_{\circledast}$. In this case, the calculation accounts for the m elements of $c_* = \tau_{\blacktriangleleft}(q)$, but ignores the $n - m$ non-zero elements, inevitably introducing an invalid result.

In both cases the reverse mapping is problematic for various reasons. In the holonomic case, one major restriction is the aforementioned (see Section 3.6) property of the matrix logarithm not necessarily being unique, i.e there exist elements $q \in \mathcal{Q}^n$ for which $\log(q)$ does not provide a unique solution. A way to ensure Eq. (4.38) being valid is to limit its domain to a subset of $\mathcal{Q}^n$ for which holds that the corresponding Lie algebra elements are within close proximity to the identity of the group, i.e., in a sufficiently small neighborhood around $T_{id}\mathcal{Q}^n$. Another limitation stems from the fact that both $C_{\circledast}$, as the image of $C_{\odot}$ under τ_*, and the corresponding envelope $C_{\circledcirc}$, are only subsets of the extended control space C_*. Even in the holonomic case, where C and C_* coincide, and even if the matrix logarithm for a given $q \in C_*$ is unique, it is not guaranteed

that the control $\tau_{\blacktriangleleft}(q)$ associated to a given configuration $q \in \mathcal{Q}^n$ is an element of $C_{\circledast}$ or even an element of $C_{\circledcirc}$. Intuitively, this means that situations exist where from a starting configuration $q_i \in \mathcal{Q}^n$ certain configurations $q_j \in \mathcal{Q}^n$ lie on a geodesic, but are 'without reach', i.e., can not directly be reached by applying a single control $c \in C_{\odot}$.

In the nonholonomic case the major difficulty stems from the fact that that the image of $C_{\odot}$ under τ_* is $C_{\circledast}$, which is only a subset of C_*. Therefore, there exist controls $c \in C_*$ that may be contained within the envelope set $C_{\circledcirc}$ but not lie on $C_{\circledast}$. This corresponds to the scenario depicted in Section 3.7, where certain configurations are only reachable by executing successive non-commutative controls. Calculating these controls, i.e., performing motion planning for nonholonomic systems, is a difficult task that would exceed the scope of this thesis. However, a good introduction to this topic can be found in [Choset et al., 2005] and [Bloch et al., 2007].

4.3.4 Sample Set

Similar to the actuator as a component that allows the entity to act, it is equipped with sensors, components that allow to retrieve information from the environment. To that end we define the entity as being equipped with $j \in \mathbb{N}_0$ individual sensors given as the j-tuple $\mathcal{P}$ with

$$\mathcal{P} = (\, p_1, p_2, \ldots, p_j \,). \tag{4.43}$$

As it holds that $j \in \mathbb{N}_0$, it is valid for an agent to have no sensors. In that case the only information the agent can perceive is provided by the partial transition function with proportional feedback (see Eq. (4.33)). We evaluate a corresponding scenario in Section 7.4.5.6.

Invoking a given sensor p_i yields a configuration- and sensor-specific perception $\tilde{\varpi}_i$, which we model as an r_i-dimensional real vector. The set of all perceptions that can be perceived by a given sensor p_i is denoted as the r_i-dimensional perception set $\mathcal{I}_i$, where due to it being composed of certain r_i-dimensional vectors from $\mathbb{R}$ it holds that

$$\mathcal{I}_i \subseteq \mathbb{R}^{r_i}. \tag{4.44}$$

As established in Section 4.3, we aim for modeling the information the entity can perceive at a given configuration $q \in \mathcal{Q}^n$ as a k-dimensional real vector, which we denote a as sample s. Thus, we define the set of all samples as the subset of $\mathbb{R}^k$ with

$$\mathcal{S} \subseteq \mathbb{R}^k. \tag{4.45}$$

As each perception set is a subset of $\mathbb{R}^{r_i}$, i.e., $\forall i \in [1, j] : \mathcal{I}_i \subseteq \mathbb{R}^{r_i}$, it holds that the set of all samples $\mathcal{S} \ni s$ can be constructed as the Cartesian product of all individual perception sets with

$$\mathcal{S} = (\mathcal{I}_1 \subseteq \mathbb{R}^{r_1}) \times (\mathcal{I}_2 \subseteq \mathbb{R}^{r_2}) \times \cdots \times (\mathcal{I}_j \subseteq \mathbb{R}^{r_j}), \tag{4.46}$$

which leads to

$$\mathcal{S} \subseteq \mathbb{R}^{r_1 \times r_2 \times \cdots \times r_j} = \mathcal{S} \subseteq \mathbb{R}^k \quad \text{with} \quad k = \sum_{i=1}^{j} r_i, \tag{4.47}$$

which allows us to model elements $s \in \mathcal{S}$ as k-dimensional real vectors. The next step is to define the relation between a configuration $q \in \mathcal{Q}^n$, elements $\tilde{\varpi}_i \in \mathcal{I}_i$, and ultimately $s \in \mathcal{S}$ by establishing the perception function π_i and the sample function Π, respectively.

4.3.5 Percept and Sample Function

As established in Section 4.3.1, the configuration space $\mathcal{Q}^n$ defines the entity's environment with respect to its topology (see Section 3.2), potential obstacles (see Eqs. (3.10) to (3.12)), and, in conjunction with the control space C, the effect of controls $c_\odot \in C_\odot$ with respect to configuration changes (see Eq. (4.11)). However, to enable the entity to perceive information about the current configuration, we have to (a) model the relation between the configuration and such information and (b) provide means for the entity to retrieve it. As stated in Section 4.2, we assume the environment to be static, which translates to the information to be perceived at any given $q \in \mathcal{Q}^n$ being static as well.

As a prerequisite, we relate each individual perception set $\mathcal{I}_i$ that is associated to the sensor p_i to the configuration space $\mathcal{Q}^n$. As elements $\varpi_i \in \mathcal{I}_i$ are the result of invoking the sensor p_i at a given configuration $q \in \mathcal{Q}^n$, we model the relation as the sensor-specific perception function

$$\pi_i : \mathcal{Q}^n \to \mathcal{I}_i \quad \text{with} \quad \pi_i(q) = \varpi_{(q,i)} \quad \text{and} \quad \varpi_{(q,i)} \in \mathcal{I}_i. \tag{4.48}$$

We account for noisy perceptions by introducing for each sensor p_i an r_i-dimensional noise vector denoted by $\epsilon_{(p,i)}$, which is sampled from a zero-mean normal distribution with sensor-specific covariance matrix $\Sigma^{p_i}_{r_i \times r_i}$, thus

$$\epsilon_{(p,i)} \sim \mathcal{N}(\, 0_{r_i}, \Sigma^{p_i}_{r_i \times r_i}), \tag{4.49}$$

where 0_{r_i} denotes the r_i-dimensional zero vector associated to the sensor p_i. The potentially noisy equivalent $\tilde{\varpi}_i$ for a given perception $\varpi_i \in \mathcal{I}_i$ is then given by the corresponding noisy perception fundtion

$$\tilde{\pi}_i : \mathcal{Q}^n \to \mathcal{I}_i \quad \text{with} \quad \tilde{\pi}_i(q) = \tilde{\varpi}_{(q,i)} \quad \text{and} \quad \tilde{\varpi}_{(q,i)} \in \mathcal{I}_i, \tag{4.50}$$

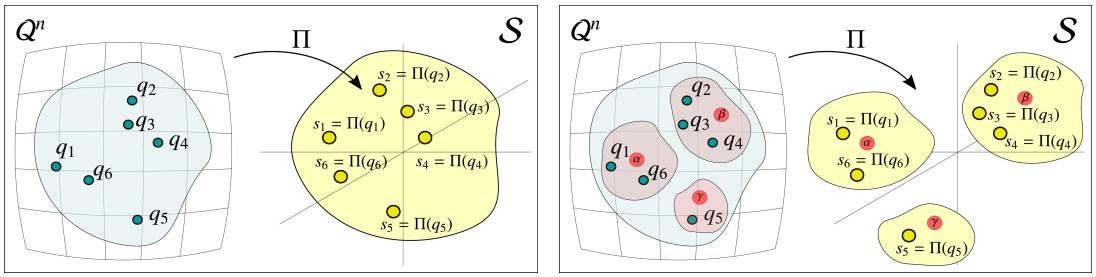
where $\tilde{\varpi}_{(q,i)}$ with respect to Eq. (4.48) is given by

$$\begin{aligned} \tilde{\pi}_i(q) &= \pi_i(q) + \epsilon_{(p,i)} \\ &= \varpi_{(q,i)} + \epsilon_{(p,i)} \\ &= \tilde{\varpi}_{(q,i)} \end{aligned} \tag{4.51}$$

In the non-noisy case, i.e., when $\Sigma^{p_i}_{r_i \times r_i}$ is set to zero, it naturally holds that $\pi_i(q) = \tilde{\pi}_i(q)$ and therefore $\tilde{\varpi}_{(q,i)} = \varpi_{(q,i)}$. Contrary to the case for noisy controls (see Eq. (4.21)), where $\tilde{c}_\odot$ is not necessarily contained in $C_\odot$, we define that $\forall i \in [1, j] : \tilde{\varpi}_i \in \mathcal{I}_i$, i.e., for each sensor it holds that all noisy perceptions are within its individual perception set. This is reasonable to assume as a perception set corresponds to the measurement range of a given sensor, which, naturally, should cover perceptions affected by noise.

We now relate the configuration space $\mathcal{Q}^n$ to the sample set $\mathcal{S}$ defined in Eq. (4.45) by defining the sample function Π that assigns to every $q \in \mathcal{Q}^n$ a sample $s \in \mathcal{S}$ with

$$\Pi : \mathcal{Q}^n \to \mathcal{S}. \tag{4.52}$$



(a) Case where the sample set $\mathcal{S}$ is connected.

(b) Case where the sample set $\mathcal{S}$ is not connected. However, local patches on $\mathcal{Q}^n$ are mapped to corresponding patches on $\mathcal{S}$.

Figure 4.5: Depiction of the sample function Π and the relation between the configuration space $\mathcal{Q}^n$ and the sample set $\mathcal{S}$. For each $q \in \mathcal{Q}^n$ there exists an element $s \in \mathcal{S}$ with $s = \Pi(q)$. Depiction of connected and unconnected sample set $\mathcal{S}$.

As $\mathcal{S}$ corresponds to the Cartesian product of the individual perception sets (see Eq. (4.46)), Π can naturally be expressed as being composed of the individual noisy perception functions Eq. (4.50), and thus it holds that

$$\Pi : \mathcal{Q}^n \rightarrow \mathcal{I}_1 \times \mathcal{I}_2 \times \cdots \times \mathcal{I}_j, \tag{4.53}$$

which corresponds to

$$(\tilde{\pi}_1 \times \tilde{\pi}_2 \times \cdots \times \tilde{\pi}_j) : \mathcal{Q}^n \rightarrow \mathcal{I}_1 \times \mathcal{I}_2 \times \cdots \times \mathcal{I}_j. \tag{4.54}$$

Therefore, a sample $s \in \mathcal{S}$ perceived at configuration $q \in \mathcal{Q}^n$ is given as

$$\begin{aligned} \Pi(q) &= (\tilde{\pi}_1 \times \tilde{\pi}_2 \times \cdots \times \tilde{\pi}_j)(q) \\ &= (\,\tilde{\pi}_1(q),\, \tilde{\pi}_2(q),\, \ldots,\, \tilde{\pi}_j(q)\,) \\ &= (\,\tilde{\varpi}_1^T,\, \tilde{\varpi}_2^T,\, \cdots,\, \tilde{\varpi}_j^T\,) \\ &= s, \end{aligned} \tag{4.55}$$

where $\forall i \in [1, j] : \tilde{\varpi}_i \in \mathcal{I}_i$, and thus s corresponds to the j-tuple of all individual perceptions, which can be represented as a k-dimensional vector. Figure 4.5a depicts the relation between $\mathcal{Q}^n$ and $\mathcal{S}$. As established in Section 4.2, we assume the relation between the configuration space $\mathcal{Q}^n$ and the sensor measurements $s \in \mathcal{S}$ to be locally smooth, which translates to the assumption that for local patches in $\mathcal{Q}^n$, a smooth mapping into $\mathcal{S}$ exist. Thus, we assume that for a local neighborhood on $\mathcal{Q}^n$ there exists a corresponding local patch on $\mathcal{S}$. In Chapter 6, we utilize this assumption to infer the functional relation between both patches using nonparametric regression algorithms. Figure 4.5b visualizes this assumption, while locally modeling Π will be discussed in Section 6.2.2.

4.4 Intrinsic View

In this section, we propose an intrinsic view on the sensorimotor system, which accounts for the fact that the agent can not directly observe its environment. Instead, it can only experience its properties by interacting with it utilizing its sensors and actuator. Instead of viewing the sensorimotor system from the point of view of an external and omniscient observer, we shift to an agent-centric one. Thus, we express sensorimotor interaction between the agent and its environment with respect to the functions we established in the last section and adapt the agent control loop (see Algorithm 1) introduced in Section 1.1 accordingly. This modified version serves as a starting point for establishing the sensorimotor map in the next chapter. This intrinsic, i.e., agent-centric view of the environment has been used in comparable approaches within the domain of bootstrapping, where [Kuipers and Levitt, 1988] states that the *"sensorimotor world of an agent, in this case, a traveler in a fixed environment, is a purely egocentric description of sensory input and motor output and contains no references to fixed features of the external environment"*. However, we do not make any assumptions with respect to the structure of the sensorimotor map, which means that we have to model the interaction with it as abstractly as possible. Thus, the intrinsic view and the correspondingly modified agent control loop can be seen as an intermediate layer between the outside world, the environment the agent interacts with using its sensors and actuators, and the sensorimotor map, the internal representation of said outside world. This conceptual view is visualized in Fig. 4.6, where the interaction with the environment corresponds to invoking the functions PERCEPT() and ACT(), and likewise the interaction with the sensorimotor map is captured by the functions INIT(), UPDATE(), and DELIBERATE(). We now define these functions and then propose an updated version of the agent control loop given in Algorithm 1

4.4.1 Sensor Invocation

Utilizing the sensors from within the agent control loop corresponds to invoking the function PERCEPT(), given in pseudocode as

$$s_i \leftarrow \text{PERCEPT}(), \tag{4.56}$$

which applies Eq. (4.52) to the current configuration $q_i \in Q^n$ and returns the corresponding sample $s_i \in \mathcal{S}$. Naturally, the current configuration q_i is not directly observable by the agent. Invoking PERCEPT() from within the agent control loop only requires information on the structure of $\mathcal{S}$ to be available. Information about the Lie group structure of the configuration space, the sample function, or the individual noise parameters are not required for invoking Eq. (4.56).

4.4.2 Actuator Invocation

We model invocation of the actuator from within the agent control loop as the function ACT(), given in pseudocode as

$$c_i^{\otimes} \leftarrow \text{ACT}(c_i^{\ominus}) \tag{4.57}$$

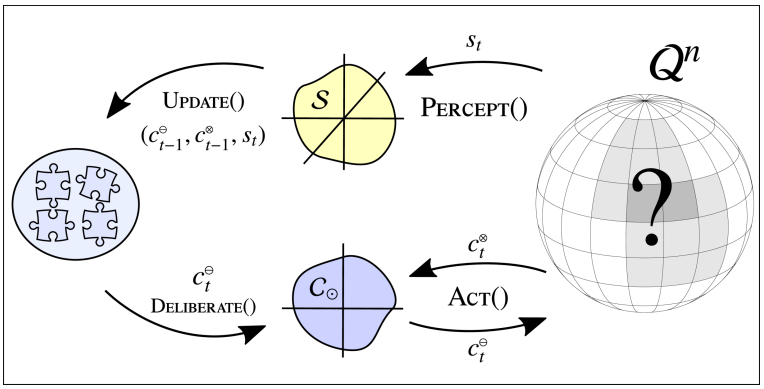


Figure 4.6: Intrinsic view of the sensorimotor system. The function PERCEPT() internally utilizes Π and allows the agent to perceive information on its environment and provides the sample $s_t \in \mathcal{S}$. The function ACT() internally utilizes $\alpha^{\triangleleft}_{\otimes}$, allows the agent to change its configuration by executing the control $c^{\ominus}_t \in C_{\odot}$, and returns proportional feedback as $c^{\otimes}_t$. The sensorimotor map is initialized with the initial sensor measurement s_1 utilizing the function INIT(). The function UPDATE() allows the agent to update the sensorimotor map by inserting new information describing its experiences of its interaction with the environment. This corresponds to the feedback ACT() provided on the last action, and the sample perceived at the new configuration which is the return value of PERCEPT(). The function DELIBERATE() utilizes the sensorimotor map to generate the next control to be executed.

which takes as argument a scheduled control

$$c^{\ominus}_i \in C_{\odot} \tag{4.58}$$

and applies it to the current configuration $q_i \in Q^n$, where the corresponding state transition from q_i to q_{i+1} that accounts for obstructions in Q^n is performed according to the partial transition function with proportional feedback $\alpha^{\triangleleft}_{\otimes}$ specified in Eq. (4.33). The function afterwards yields feedback on the degree of application of the scheduled control, denoted the effective control $c^{\otimes}_i \in C_{\odot}$, which is given as

$$c^{\otimes}_i = t^{\otimes}_i c^{\ominus}_i. \tag{4.59}$$

According to Eqs. (4.28) and (4.29), the scaling factor $t^{\otimes}_i$ is based on q_i and $c^{\ominus}_i$ with

$$t^{\otimes}_i = (\mathrm{free}_{\otimes} \circ \mathrm{path})(q_i, c^{\ominus}_i). \tag{4.60}$$

The effective control $c^{\otimes}_i$ is a scaled version of the scheduled control $c^{\ominus}_i$ and provides the required odometry feedback as discussed in Section 4.2. Naturally, neither the configuration q_i nor q_{i+1} are directly observable by the agent. Invoking ACT() from within the agent control loop only requires information on the structure of $C_{\odot}$ to be available. Information about the topology of Q^n, i.e., information about the Lie group structure of the configuration space, the parameters of the transition function $\tau_{\blacktriangleright}$, or the noise ϵ_c are not required for invoking Eq. (4.57).

4.4.3 Invoking the Sensorimotor Map

The sensorimotor map has to provide functions for initialization, a way of updating it with new information to maintain a consistent state, and a method that provides the next control to be executed. It has to be noted, that the interface between the agent loop and the sensorimotor map differs from the one proposed in the generic agent loop given in Algorithm 1. These changes, to the order of invocations, as well as to the signature of the functions, are required for the development of the sensorimotor map algorithms in Chapter 5 and the bootstrapping algorithms in Chapter 6. Initializing the sensorimotor map corresponds to invoking the function INIT(), given as

$$map \leftarrow \textsc{init}(s_1, Q^n, T_{\circlearrowleft}, T_{\circlearrowright}), \tag{4.61}$$

where $s_1 \in \mathcal{S}$ denotes the first sample perceived by the initial invocation of PERCEPT(). However, this function is special, as it requires as an argument the Lie group representing the configuration space Q^n as well as the matrices $T_{\circlearrowleft}$ and $T_{\circlearrowright}$ that encode the effect controls have on the configuration. This might seem counterintuitive with respect to our aim of developing an algorithm for the purpose of detecting exactly these properties. The reason behind this parameterization is that we initially assume to know Q^n, $T_{\circlearrowleft}$, and $T_{\circlearrowright}$, which allows us to formally establish the sensorimotor map as a manifold-based graph structure in Chapter 5. However, in Chapter 6 we then utilize the sensorimotor map for bootstrapping, i.e., as an objective function for evaluating hypotheses about Q^n, $T_{\circlearrowleft}$ and $T_{\circlearrowright}$.

Utilizing the sensorimotor map from within the control loop for storing information on control execution corresponds to invoking the function UPDATE(), given in pseudocode as

$$map \leftarrow \textsc{udpate}(c^{\ominus}_{i-1}, c^{\otimes}_{i-1}, s_i), \tag{4.62}$$

where the 3-tuple $(c^{\ominus}_{i-1}, c^{\otimes}_{i-1}, s_i)$ encodes the feedback ACT() provided on the last action and the sample PERCEPT() returned at the new configuration.

Utilizing the sensorimotor map for generating the next control corresponds to invoking the function DELIBERATE(), given in pseudocode as

$$c^{\ominus}_i \leftarrow \textsc{Deliberate}(\, map\,), \tag{4.63}$$

where $c^{\ominus}_i$ is scheduled to be provided as an argument to ACT() and marks the planned transition from time step i to $i+1$. As the focus of this thesis is set on developing the sensorimotor map and utilizing it for bootstrapping, we implemented a suitable low-level behavior where the control sequences generated by DELIBERATE() are inspired by the concept of motor babbling (see Section 2.2.3). The motor-babbling approach is discussed in Section 6.4 where Section 7.2.4 will cover details on the patterns used in the evaluation.

4.4.4 Intrinsic Agent Loop

The intrinsic agent loop given in Algorithm 2 is a modified version of the agent loop introduced in Algorithm 1. It models the interaction with the environment utilizing Eq. (4.56), Eq. (4.57), and Eq. (4.61), and the interaction with the sensorimotor map utilizing Eq. (4.62) and Eq. (4.63).

Intrinsic Agent Loop

Require: $Q^n, T_{\cup}, T_{\cup}$ given
1: **procedure** IntrinsicAgentLoop
    ▷ Initialize time index
2: $t \leftarrow 1$
    ▷ Perform initial percept
3: $s_t \leftarrow$ percept() ▷ Eq. (4.56)
    ▷ Initialize internal representation
4: $map \leftarrow$ Init(s_t, Q^n, $T_{\cup}$, $T_{\cup}$) ▷ Eq. (4.61)
    ▷ Execute main loop as long as the agent is active
5: **while** Active() **do**
    ▷ Deliberate on next control
6: $c_t^{\ominus} \leftarrow$ Deliberate(map) ▷ Eq. (4.63)
    ▷ Invoke actuator to perform state transition
7: $c_t^{\otimes} \leftarrow$ Act($c_t^{\ominus}$) ▷ Eq. (4.57)
    ▷ Increase time index to indicate state transition
8: $t \leftarrow t + 1$
    ▷ Perform percept at new configuration
9: $s_t \leftarrow$ Percept() ▷ Eq. (4.56)
    ▷ Update internal representation
10: $map \leftarrow$ Update($c_{t-1}^{\ominus}$, $c_{t-1}^{\otimes}$, s_t) ▷ Eq. (4.62)
11: **end while**
12: **end procedure**

Algorithm 2: Intrinsic agent loop. It models an intermediate layer between the environment and the sensorimotor map. It is created as a modification of the agent loop given in Algorithm 1. Modifications were required to utilize the algorithm in Chapter 5 and Chapter 6. Interaction with the environment is modeled by the functions percept() and act(). Interaction with the sensorimotor map is modeled by the functions init(), update(), and deliberate().

4.5 Summary

In this chapter, we established a formal representation of the interaction between a mobile agent and its environment with respect to the agent's sensorimotor properties. We related its motor controls to the tangent space of the matrix Lie group that represents its configuration space, thus allowing a parameterizable mapping from control vectors to group transformations. Similarly, we related the configuration space to the set of samples, modeling the information the agent can perceive through its sensors. Utilizing these relations, we established two distinct views on the sensorimotor system.

The extrinsic view models the interaction between the agent and the environment as viewed from an external observer defining the functional relation between the agent motor controls, its configuration space, and its sensory inputs.

The intrinsic view accounts for the limited knowledge the agent has, modeling only those properties immediately observable by the agent and thus providing a domain-independent intermediate layer between the external world and its internal representation.

In the next chapter, we utilize both these views to formally establish the sensorimotor map as a manifold-based graph structure, while in Chapter 6 we utilize it for bootstrapping.

5
Sensorimotor Maps

In this chapter, we utilize the definitions from Chapters 3 and 4 to develop the sensorimotor map, a graph-based joint sensorimotor structure that establishes a tightly-coupled connection between the topology of the configuration space and the sensory and motor capabilities of the agent. The structure is as follows: In Section 5.2, we lay the foundation for the development of the sensorimotor map by introducing its two central building blocks, the sensorimotor chain and the sensorimotor neighborhood. In Section 5.2.1, we formally define the former as a sequence of sensory inputs and motor controls. It serves as a hypothesis- and domain-independent representation of the agent's sensorimotor interaction with its environment. In Section 5.2.2, we define sensorimotor neighborhoods as local vector spaces that relate patches from the configuration space to the agent-specific control space as well as its sample set, ultimately representing the perceived sensor measurements from local patches of the environment in agent- and thus motor-specific reference frames. Both concepts are combined in Section 5.2.3, where we connect sensorimotor neighborhoods to the sensorimotor chain, ultimately allowing us to define transformations between neighborhoods and to represent them with respect to arbitrary reference frames attached to the indirectly observed configurations the agent has traversed through. In Section 5.3.2, we build upon these foundations and propose an initial sensorimotor representation structure denoted as the sensorimotor graph. We refine this approach in Section 5.3.3 by developing an optimized version denoted as the sensorimotor map, which establishes an evenly-spaced sensorimotor tessellation of the configuration space, reducing the number of required reference frames drastically. The chapter concludes with a summary and an outlook towards the next chapter in Section 5.4.

5.1 Synopsis

In this chapter, we utilize both the extrinsic definition of a sensorimotor system introduced in Section 4.3 and the intrinsic view of the agent given in Section 4.4 to create a joint sensorimotor representation denoted as the sensorimotor map. This structure relies on two fundamental building blocks: the sensorimotor chain and the sensorimotor neighborhood.

We establish the sensorimotor chain by building upon the sequential interaction of the agent with its environment defined by the intrinsic agent loop and focus on the information available in each step, i.e., on the information provided by the sensors and the actuators. Thus, we define the sensorimotor chain as an alternating sequence of samples as well scheduled and effective controls. This sequence is a domain-independent and topology-invariant representation of the sensorimotor interaction between the agent and its environment, i.e., a low-level data stream that documents all information available to the agent at a given time. Moreover, this alternating sequence encodes a path in the configuration space, as each motor control corresponds to a transition from one configuration to another, while the samples denote the information that is perceivable at the respective configurations.

We then establish motor neighborhoods that are constructed by creating a set of tuples, where elements from the agent's control space are paired with their corresponding matrix Lie group transformations. This motor neighborhood establishes a reference frame, i.e., a structure that associates motor controls with distinct configurations from the configuration space. By applying elements of this set to a given configuration, we establish a local neighborhood on the configuration space that is entirely specified by the motor capabilities of the agent. The sensorimotor neighborhoods are then established by applying the sample and occupancy functions to these configurations, effectively creating local reference frames that associate motor controls to perceivable samples and obstruction information, respectively. As a final step, we show how elements from one neighborhood can be represented with respect to another, i.e., how reference frame transformations can be calculated.

We then combine both concepts, the sensorimotor chain as a sequence of motor controls that correspond to configuration changes and the sensorimotor neighborhoods as motor based reference frames that represent the samples and obstruction within a given patch of the configuration space. To this end, we attach a sensorimotor neighborhood to each configuration from the sensorimotor chain, where transitions between these neighborhoods are given by the corresponding effective controls. The basic concept is thus not to utilize a global map as an extrinsic reference frame, but instead to represent changes in sensor measurements in relation to the executed motor controls and 'patch' together local agent-specific reference frames where these frames are based solely on the dimension and structure of the entity's motor controls. By attaching a neighborhood to each configuration encoded by the sensorimotor chain, we effectively create a graph structure as a set of overlapping neighborhoods that locally represent the configuration space. We optimize this sensorimotor graph structure and reduce the number of neighborhoods by selecting only a subset of the memory nodes as reference frame centers, aiming for a evenly-spaced tessellation of the configuration space, which we denote as the sensorimotor map. This representation is tailored to the agent with respect to its sensor capabilities, its motor capabilities, and its history of sensorimotor interaction. It has to be noted that within this chapter we

assume that the matrix Lie group that represents the configuration space is known, which also holds for the mapping that associates controls to tangent space elements. Thus, we assume the correct information, i.e., the geometric interpretation of the motor controls to be available in order to initially establish the sensorimotor map representation. However, in the next chapter we change the scenario and show how the sensorimotor map can be utilized as an objective function for discovering the combination of the matrix Lie group representing the configuration space and the matrices encoding the effect of motor controls.

5.2 Data Structures and Topological Relations

In this section we cover the two fundamental building blocks of the sensorimotor map. In Section 5.2.1 we establish the sensorimotor chain as the intrinsic representation of the agent's sensorimotor interaction with its environment. In Section 5.2.2 we propose sensorimotor neighborhoods as agent-specific, i.e., motor-specific reference frames. We show how to perform reference frame transformations and discuss the geometric implications, given in Section 5.2.3.

5.2.1 The Sensorimotor Chain

We start by modeling the information that is available to the agent based on the intrinsic agent loop given in Algorithm 2. Successive invocation of said loop corresponds to alternately executing the functions PERCEPT() (see Eq. (4.56)) and ACT() (see Eq. (4.57)). This ultimately creates an alternating sequence of samples s_i, scheduled controls $c_i^{\ominus}$, and the corresponding effective controls $c_i^{\otimes}$. As these two functions are the only interface between the agent and the outside world, the samples $s_i \in \mathcal{S}$ provided by Eq. (4.56) and the feedback on control execution, i.e., the relation between scheduled and effective controls $c_i^{\ominus}, c_i^{\otimes} \in C_{\odot}$, provided by Eq. (4.57), is the closest thing to ground truth data the agent can have of its combined sensory and motor interaction with its environment. Thus, every conclusion about the topology of the configuration space $\mathcal{Q}^n$, the causal relation between controls $C_{\odot}$ and $\mathcal{Q}^n$, and the nature of the relation between configurations $\mathcal{Q}^n$ and perceivable samples from $\mathcal{S}$ has to, and ultimately can only be, based on this data. We therefore utilize this information, i.e., the aforementioned sequence of samples and controls, as a foundation for building the sensorimotor map. In the following, the index i indicates that an element is associated to an arbitrary time step. The index t indicates that an element is the last element of a given sequence, i.e., that it is associated to the current time step.

We denote the sequence of samples and controls as the sensorimotor sequence of length t with L_t given by

$$L_t = (\, s_1, (c_1^{\ominus}, c_1^{\otimes}), s_2, (c_2^{\ominus}, c_2^{\otimes}), s_3, \ldots, s_t \,), \tag{5.1}$$

where $\forall i \in [1, t] : s_i \in \mathcal{S}$ and $\forall i \in [1, t-1] : c_i^{\ominus}, c_i^{\otimes} \in C_{\odot}$. Naturally, L_t terminates with s_t, which corresponds to the sample perceived at the latest, i.e., current, configuration of the agent. As each sample $s_i \in \mathcal{S}$ corresponds to a distinct configuration $q_i \in \mathcal{Q}^n$ of the agent at the point of invoking Π (see Eq. (4.52)), the controls naturally encode the transitions between consecutive configurations $q_i, q_{i+1} \in \mathcal{Q}^n$. A visualization of L_t as a sequence of connected paths on $\mathcal{Q}^n$ is given in Figure 5.1.

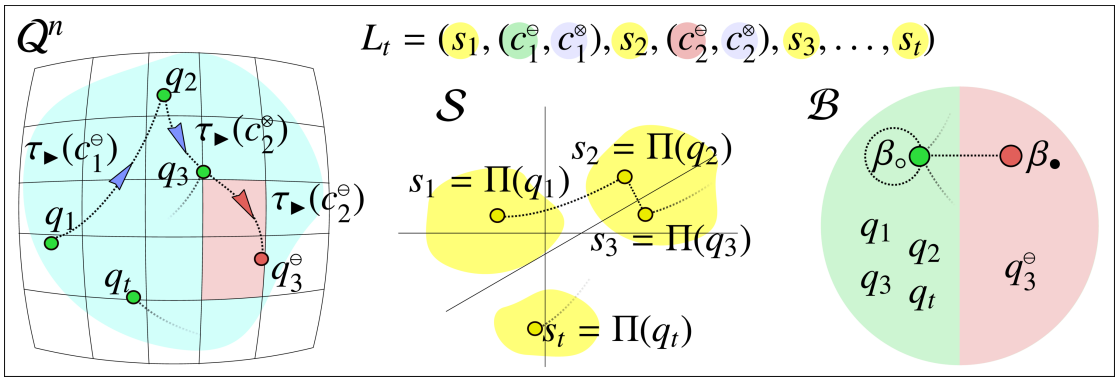


Figure 5.1: Visualization of the sensorimotor sequence L_t. It encodes the history of the sensorimotor interaction between the agent and its environment. Every sample s_i encodes the information perceived at the corresponding configuration q_i as given by $s_i = \Pi(q_i)$. Each indirectly observed configuration q_i the agent traversed through is unobstructed and thus associated with $\beta_\circ \in \mathcal{B}$. The effective control $c_i^\otimes$ encodes the transition between the subsequent configurations q_i and q_{i+1} where q_{i+1} is created as $q_{i+1} = q_i \bullet \tau_\blacktriangleright(c_i^\otimes)$. If the scheduled control $c_i^\ominus$ differs from $c_i^\otimes$, the path between q_i and $q_i \bullet \tau_\blacktriangleright(c_i^\ominus)$ moves through a section on $\mathcal{Q}^n$ that is obstructed (depicted as red squares). This is the case for q_2, where the scheduled and effective controls $c_2^\ominus$ and $c_2^\otimes$ differ, and the path from q_2 to $q_2^\ominus$ is obstructed. Here, $q_2^\ominus$ denotes the configuration the agent would reach if executing $c_2^\ominus$ at q_2 was possible. To indicate that $c_2^\ominus$ can not be executed, $q_2^\ominus$ is associated with $\beta_\bullet \in \mathcal{B}$.

We can view L_t as being composed of $t-1$ sensorimotor links given as the 3-tuples

$$l_i = (\, s_i, (c_i^\ominus, c_i^\otimes), s_{i+1} \,) \quad \text{with} \quad i \in [1, t-1], \tag{5.2}$$

consisting of the sample $s_i \in \mathcal{S}$, perceived at configuration $q_i \in \mathcal{Q}^n$ in time step i, the sample $s_{i+1} \in \mathcal{S}$, perceived at configuration $q_{i+1} \in \mathcal{Q}^n$ in time step $i+1$ and the tuple of scheduled and effective controls $(c_i^\ominus, c_i^\otimes) \in C_\odot$ that encodes the transition between the configurations q_i and q_{i+1} and thus between the perceived samples s_i and s_{i+1}. This view sets the focus on the transitions between perceived samples and relates two consecutive samples to each other in terms of the control that connects them. Sensorimotor links, as atomic elements, can be compared to the sensorimotor features proposed in [Zetzsche et al., 2008] that capture the relation between two sensory inputs perceived at different configurations with respect to a motor action that encodes the corresponding state transition. A similar concept are the actions described in [Kuipers and Levitt, 1988] that correspond to *"a change of state, with the current view defined immediately before and after an action but not changing continuously during it"*, where a view *"represents the travelers sensory input at a given instant"*. With respect to these approaches, however, a difference lies in the way motor controls are modeled. Instead of explicitly encoding actions by relating them to their geometric meaning, we implicitly encode configuration changes by only providing the associated control vectors, effectively separating control output and actual geometric interpretation from each other. The sensorimotor sequence is thus devoid of any geometric interpretation and solely depicts the sensorimotor interaction as it has been perceived and thus can be denoted the 'subjective ground truth' the agent has of its sensorimotor interaction with its environment. This is in line with [Kuipers and Levitt, 1988] where views and actions *"are*

assumed to be correct descriptions of the actual experience, although possibly quite abstracted and incomplete".

Within the context of this thesis we propose a different view on the sequence L_t, as we shift the focus from the transitions $(c_i^{\ominus}, c_i^{\otimes})$ to individual samples s_i and the indirectly observed configurations q_i they are associated to. Thus, we interpret L_t as the consecutive list of t memory nodes $\omega_1, \cdots, \omega_t$, where each memory node ω_i indirectly encodes the distinct configuration $q_{(\omega,i)} \in \mathcal{Q}^n$ through the sample s_i associated to it. To each memory node ω_i, we associate the effective control $-c_{i-1}^{\otimes}$ that describes the transition to its predecessor ω_{i-1} and the effective control $c_i^{\otimes}$ that describes the transition to its successor ω_{i+1}. Naturally the first node ω_1 has no predecessor and the last node ω_t has no successor. Here, $c_{t-1}^{\otimes}$ has to be negated, as it originally corresponds to the transformation from $q_{(\omega,i-1)}$ to $q_{(\omega,i)}$, i.e., the transformation from ω_{i-1} to ω_i. Thus, the reverse transformation from ω_i to its predecessor ω_{i-1} is given by $-c_{i-1}^{\otimes}$.

In addition to the effective controls $-c_{i-1}^{\otimes}$ and $c_i^{\otimes}$ we associate the scheduled control $c_i^{\ominus}$ to each node , where the relation between scheduled and effective controls is given by Eq. (4.59) as $c_i^{\otimes} = t_i^{\otimes} c_i^{\ominus}$. If $c_i^{\ominus} \neq c_i^{\otimes}$ it holds that the proportional feedback $t^{\otimes}$ is lower than 1. This represents the case where the path between q_i and $q_i \bullet \tau_{\blacktriangleright}(c_i^{\ominus})$ is obstructed, i.e., q_i is a configuration at which the scheduled control $c_i^{\ominus}$ can not be executed without encountering an obstacle. Attaching $c_i^{\ominus}$ to ω_i thus allows us to encode properties on occupied portions of the configuration space by representing them in terms of obstructed path segments.

An intermediate memory node ω_i created from a section of the sensorimotor sequence L_t is therefore given as the tuple

$$\omega_i = \left\{ \begin{array}{ll} (\{-c_{i-1}^{\otimes}\}, \{s_i\}, \{c_i^{\ominus}, c_i^{\otimes}\}), & \text{if} \quad c_i^{\ominus} \neq c_i^{\otimes} \\ (\{-c_{i-1}^{\otimes}\}, \{s_i\}, \{c_i^{\otimes}\}), & \text{otherwise} \end{array} \right\} \quad \text{if} \;\; t > 2 \;\; \text{and} \;\; 1 < i < t. \tag{5.3}$$

Naturally, the first node ω_1 has no predecessor, thus it holds that

$$\omega_1 = \left\{ \begin{array}{ll} (\{\}, \{s_1\}, \{c_1^{\ominus}, c_1^{\otimes}\}), & \text{if} \quad c_1^{\ominus} \neq c_1^{\otimes} \\ (\{\}, \{s_1\}, \{c_1^{\otimes}\}), & \text{otherwise} \end{array} \right\} \quad \text{if} \;\; t > 1 \;\; \text{and} \;\; i = 1. \tag{5.4}$$

Correspondingly the last node ω_t has no successor, thus

$$\omega_t = (\{-c_{t-1}^{\otimes}\}, \{s_t\}, \{\}) \quad \text{if} \;\; t > 1 \;\; \text{and} \;\; i = t. \tag{5.5}$$

A sensorimotor sequence L_1 containing only a single sample corresponds to the trivial case where

$$\omega_t = (\{\}, \{s_1\}, \{\}) \quad \text{if} \;\; t = 1 \;\; \text{and} \;\; i = 1. \tag{5.6}$$

To be able to explicitly refer to the individual components of a given memory node, i.e., the different sets containing controls and the sample, we establish the following notation, where ω_i is given as

$$\omega_i = \{ (\overleftarrow{C}[\omega_i], \mathcal{S}[\omega_i], \overrightarrow{C}[\omega_i]) \}. \tag{5.7}$$

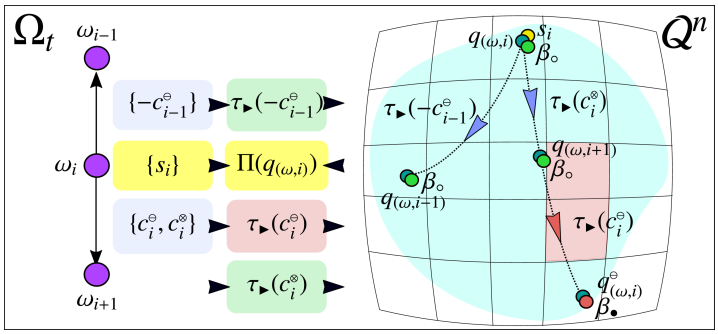


Figure 5.2: Relation between the intermediate memory node ω_i and the configuration space $\mathcal{Q}^n$. The memory node ω_i encodes the, not directly observable, configuration $q_{(\omega,i)}$. The effective control $-c^{\otimes}_{i-1}$ between ω_i and its predecessor ω_{i-1} encodes the transformation $\tau_{\blacktriangleright}(-c^{\otimes}_{i-1})$ from $q_{(\omega,i)}$ to $q_{(\omega,i-1)}$. The effective control $c^{\otimes}_i$ between ω_i and its successor ω_{i+1} encodes the transformation $\tau_{\blacktriangleright}(c^{\otimes}_i)$ from $q_{(\omega,i)}$ to $q_{(\omega,i+1)}$. However, the scheduled control $c^{\ominus}_i$ encodes the transformation $\tau_{\blacktriangleright}(c^{\ominus}_i)$ from $q_{(\omega,i)}$ to $q^{\ominus}_{(\omega,i)}$, a path that is partially obstructed. The sample s_i associated to ω_i corresponds to the feedback of the sample function Π at $q(\omega, i)$. The elements from the obstruction set $\beta_\circ, \beta_\bullet \in \mathcal{B}$, indicate free and obstructed paths, i.e.,executable and not executable controls, respectively. Notation-wise it has to be remarked that $-q$ denotes the inverse of q which, as q is a matrix, would correspond to q^{-1}. However, to avoid clutter in the superscripts, we represent the reverse operation in the above way.

According to Eqs. (5.4) to (5.6), $\overleftarrow{C}[\omega_i]$ is given as

$$\overleftarrow{C}[\omega_i] = \{\} \quad \text{or} \quad \overleftarrow{C}[\omega_i] = \{-c^{\otimes}_{t-1}\}. \tag{5.8}$$

According to Eqs. (5.4) to (5.6), $\overrightarrow{C}[\omega_i]$ is given as

$$\overrightarrow{C}[\omega_i] = \{\} \quad \text{or} \quad \overrightarrow{C}[\omega_i] = \{c^{\ominus}_1, c^{\otimes}_1\} \quad \text{or} \quad \overrightarrow{C}[\omega_i] = \{c^{\otimes}_1\}. \tag{5.9}$$

Accordingly, $\mathcal{S}[\omega_i]$ is given as

$$\mathcal{S}[\omega_i] = \{\} \quad \text{or} \quad \mathcal{S}[\omega_i] = \{s_i\}. \tag{5.10}$$

These sets are utilized in the algorithms developed in Sections 5.3.2 and 5.3.3. Figure 5.2 visualizes the relation between an intermediate memory node ω_i and the configuration space $\mathcal{Q}^n$.

We denote the consecutive sequence of t memory nodes, created from a given sensorimotor sequence L_t, as the sensorimotor chain

$$\Omega_t = (\,\omega_1, \omega_2, \omega_3, \cdots, \omega_t\,). \tag{5.11}$$

Naturally each memory node corresponds to a configuration $q \in \mathcal{Q}^n_\circ$ given by

$$q_{(\omega,i)} \in \mathcal{Q}^n_\circ, \tag{5.12}$$

where $q_{(\omega,i)}$ denotes the configuration associated to the memory node ω_i. Likewise, the sample s_i associated to $q_{(\omega,i)}$ naturally corresponds to

$$s_i = s_{(\omega,i)} = \Pi(q_{(\omega,i)}). \tag{5.13}$$

Moreover, it holds that:

a.) The sensorimotor chain Ω completely encodes the history of sensory inputs and motor controls as observed by the agent.

b.) Whether samples and controls are noise-free or subject to noise has no structural implication on the way the sensorimotor chain is constructed and represented.

c.) The sensorimotor chain is unrelated to the topological interpretation of controls or information on individual perceptions, thus making it independent of the configuration space $\mathcal{Q}^n$.

In a noise-free environment every previously observed configuration q_i can be reached by subsequently following the chain of effective controls forward or backward where, as the set of valid controls is assumed to be symmetric (see Sections 4.2 and 4.3.2), for each $c_\odot \in C_\odot$ the inverse control $-c_\odot$ is guaranteed to exist. In setting where the actuator is subject to noise, however, the situation is different, as executing the inverse of a previously executed control inevitably introduces an offset. Executing $c_\odot$ at $q_i \in \mathcal{Q}^n$ moves the agent to q_j. Executing the inverse control $-c_\odot$ moves the agent to $q_i' \in \mathcal{Q}^n$ and most likely it holds that $q_i \neq q_i'$. Executing successive noisy controls causes this offset to accumulate. Thus, in a noisy scenario, following the chain of effective controls, is not a reliable way to reach a previous configuration.

We denote the sequence of effective controls that connect two given memory nodes ω_i and ω_j by

$$C_{(\Omega,\odot)}[\omega_i, \omega_j] = \begin{cases} c_i^\otimes, c_{i+1}^\otimes, \cdots, c_{j-1}^\otimes & \text{if} \quad i < j \\ -c_{i-1}^\otimes, -c_{i-2}^\otimes, \cdots, -c_j^\otimes & \text{if} \quad i > j \\ 0_\odot & \text{if} \quad i = j, \end{cases} \tag{5.14}$$

where naturally $0_\odot \in C_\odot$ denotes the zero control from $C_\odot$.

The sensorimotor chain as a representation of the sensorimotor interaction between the agent and its environment can be related to the sensorimotor input proposed in [Kuipers and Levitt, 1988] given as an *"alternating sequence of views and actions"*. As a memory node captures the sensory input perceived at a given configuration, it can the compared to the 'distinctive states' proposed by [Kuipers, 2000]. Another related concept is given by [Buhrmann et al., 2013]. where the 'sensorimotor habitat' is proposed as the *"set of all sensorimotor trajectories"*. As the sensorimotor chain is a connected sequence of trajectories, we can relate it to this definition by interpreting it as a specific instance of such a trajectory or alternatively as a subset of multiple trajectories from this set.

In Section 5.2.2, we establish reference frames on $\mathcal{Q}^n$, which we use later to represent the sensorimotor chain.

5.2.2 Sensorimotor Neighborhoods

In this section, we establish the concept of a local reference frame based on the matrix Lie group representing $\mathcal{Q}^n$ and the set of executable controls $C_\odot$. This means that the agent utilizes its own motor capabilities as a reference frame for representing its environment.

5.2.2.1 Motor Covers and Sensorimotor Neighborhoods

As established in Eq. (4.17), the function $\tau_{\blacktriangleright}$ allows us to map elements from the control space C to elements from the configuration space $\mathcal{Q}^n$ by applying the subsequent transformations $\tau_{\blacktriangleright}$ is composed of, i.e., τ_* (Eq. (4.5)), $\tau_{\circlearrowleft}$ (Eq. (4.12)), and the exponential map (Eq. (3.62)). It holds that

$$\overbrace{C \xrightarrow{\tau_*} C_* \xrightarrow{\tau_{\circlearrowleft}} T_{id}\mathcal{Q}^n \xrightarrow{\exp} \mathcal{Q}^n}^{\tau_{\blacktriangleright} : C \to \mathcal{Q}^n}. \tag{5.15}$$

Thus, $\tau_{\blacktriangleright}$ maps individual elements from C to individual elements from $\mathcal{Q}^n$. Consequently, as $C_{\odot} \subseteq C$, this holds for elements from $C_{\odot}$ too. However, we are not interested in looking at individual elements from C or $C_{\odot}$. Instead, we consider the set of all executable controls $C_{\odot}$ as a whole. We denote $\mathcal{R}_{\odot}$ as the motor cover of $C_{\odot}$, which is the set of tuples of all elements in $C_{\odot}$ and their corresponding transformations on $\mathcal{Q}^n$ given as

$$\mathcal{R}_{\odot} = \{\, (c, q) \,|\, q = \tau_{\blacktriangleright}(c),\ c \in C_{\odot} \,\}. \tag{5.16}$$

The tuples in $\mathcal{R}_{\odot}$ are independent of the current state $q \in \mathcal{Q}^n$ of the agent, as it is entirely defined by the relation between elements from $C_{\odot}$ and the configuration space $\mathcal{Q}^n$, i.e., by the properties of $\tau_{\blacktriangleright}$. The motor cover thus provides a generic encoding of the relation between valid motor controls and the corresponding transitions on the manifold, where a visualization of $\mathcal{R}_{\odot}$ is given in Fig. 5.3a.

Applying transformations from $\mathcal{R}_{\odot}$ to an arbitrary configuration $q_i \in \mathcal{Q}^n$ yields

$$\mathcal{M}_{\odot}[q_i] = \{\, (c, q) \,|\, q = q_i \bullet q_j,\ (c, q_j) \in \mathcal{R}_{\odot} \,\} \quad \text{with} \quad q_i \in \mathcal{Q}^n, \tag{5.17}$$

which is the motor neighborhood of the configuration q_i defined with respect to the motor cover $\mathcal{R}_{\odot}$. An alternative view is given by

$$\mathcal{M}_{\odot}[q_i] = \{\, (c, q) \,|\, q = q_i \bullet \tau_{\blacktriangleright}(c),\ c \in C_{\odot} \,\} \quad \text{with} \quad q_i \in \mathcal{Q}^n, \tag{5.18}$$

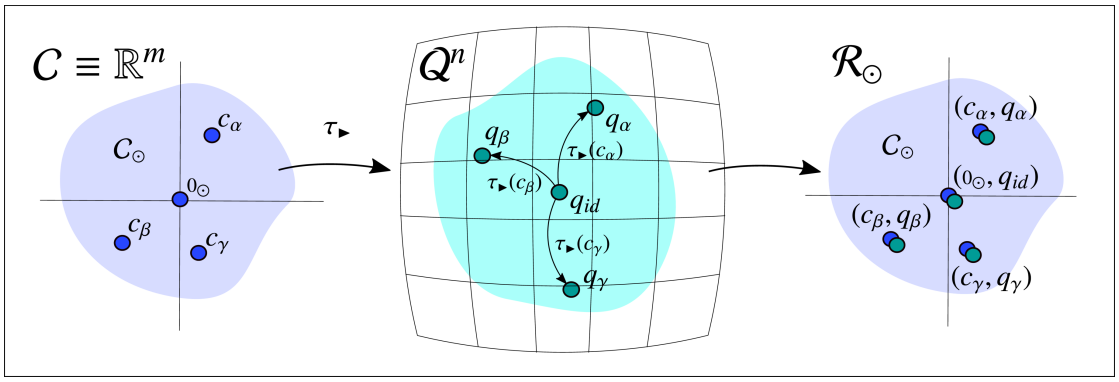
which defines the motor neighborhood of the configuration q_i solely based on Eq. (4.17) and with respect to elements from $C_{\odot}$. In both cases, however, the motor neighborhood represents the accessible patch of the configuration space $\mathcal{Q}^n$ around a distinct element q_i, where tuples $(c, q) \in \mathcal{M}_{\odot}[q_i]$ consist of a configuration q in the neighborhood of q_i as well as the corresponding control $c \in C_{\odot}$ that is required to reach it. Applying $\mathcal{R}_{\odot}$ to a distinct configuration q_i thus establishes a local reference frame with q_i as its origin, which is visualized in Fig. 5.3b.

To incorporate information on samples, i.e., perceivable properties of the configurations contained in a given motor neighborhood $\mathcal{M}_{\odot}[q_i]$, we apply the non-noisy sample function Π (see Eq. (4.52)) to its individual configurations, which yields

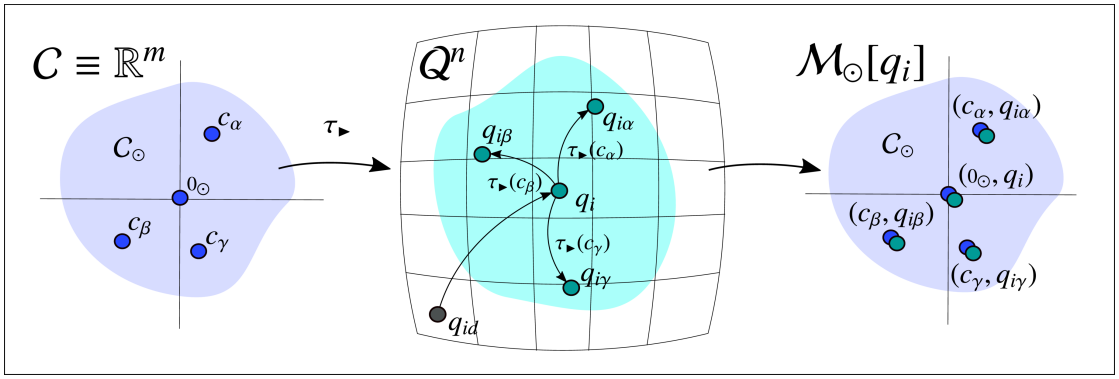
$$\Lambda_{(\mathcal{S},\odot)}[q_i] = \{\, (c, s) \,|\, s = \Pi(q),\ (c, q) \in \mathcal{M}_{\odot}[q_i] \,\} \quad \text{with} \quad q_i \in \mathcal{Q}^n. \tag{5.19}$$

the sensorimotor sample neighborhood of the configuration q_i defined with respect to the motor neighborhood $\mathcal{M}_{\odot}[q_i]$. An alternative view is given by

$$\Lambda_{(\mathcal{S},\odot)}[q_i] = \{\, (c, s) \,|\, s = \Pi(q_i \bullet \tau_{\blacktriangleright}(c)),\ c \in C_{\odot} \,\} \quad \text{with} \quad q_i \in \mathcal{Q}^n, \tag{5.20}$$



(a) Visualization of the motor cover $\mathcal{R}_\odot$. Applying $\tau_\blacktriangleright$ to elements $0_\odot$, c_α, c_β, c_γ from $C_\odot$ yields the corresponding group elements $q_{id}, q_\alpha, q_\beta, q_\gamma$. Each q_j can be seen as $q_{id} \bullet \tau_\blacktriangleright(c_j)$, where naturally $q_{id} \bullet \tau_\blacktriangleright(0_\odot) = q_{id}$. Thus, $\mathcal{R}_\odot$ consists of all tuples from the set of valid controls and their corresponding transformations.



(b) Visualization of the motor neighborhood $\mathcal{M}_\odot[q_i]$. Applying $\tau_\blacktriangleright$ to elements $0_\odot$, c_α, c_β, c_γ from $C_\odot$ yields the corresponding group elements $q_{id}, q_\alpha, q_\beta, q_\gamma$. Applying these to q_i yields q_{ij} as $q_i \bullet \tau_\blacktriangleright(c_j)$. Naturally, q_i is given as $q_i = q_{id} \bullet q_i$. Thus, $\mathcal{M}_\odot[q_i]$ is the set of all tuples (c, q), where c is an element from the set of valid controls and q is the associated configuration from Q^n, created by applying $\tau_\blacktriangleright(c)$ to the origin of the motor neighborhood q_i.

Figure 5.3: Visualization of the motor cover $\mathcal{R}_\odot$ and the motor neighborhood $\mathcal{M}_\odot[q_i]$.

which is solely based on the combined mapping from C to Q^n given by Eq. (4.17), the mapping $Q^n \times Q^n$ to Q^n given by Eq. (3.44), and the mapping from Q^n to $\mathcal{S}$ given by Eq. (4.52). According to the individual definitions, it holds that

$$\underbrace{Q^n \times \overbrace{(C \xrightarrow{\tau_*} C_* \xrightarrow{\tau_\cup} T_{id}Q^n \xrightarrow{\exp} Q^n)}^{\tau_\blacktriangleright : C \to Q^n} \to Q}_{\bullet : Q^n \times Q^n \to Q^n} \xrightarrow{\Pi} \mathcal{S}. \tag{5.21}$$

In both cases the sensorimotor sample neighborhood $\Lambda_{(\mathcal{S},\odot)}[q_i]$ is a local reference frame where tuples $(c, s) \in \Lambda_{(\mathcal{S},\odot)}[q_i]$ encode the sensorimotor properties of the neighborhood of $q_i \in Q^n$ by directly relating sensory information $s \in \mathcal{S}$ to motor controls $c \in C_\odot$. A visualization of $\Lambda_{(\mathcal{S},\odot)}[q_i]$ is given in Fig. 5.4a.

To correspondingly incorporate obstructions, i.e., information on whether a given configuration in a motor neighborhood $\mathcal{M}_\odot[q_i]$ is empty or free, we apply Eq. (3.10) to the individual

configurations, which yields

$$\Lambda_{(\mathcal{B},\odot)}[q_i] = \{\,(c,\beta)\,|\,\beta = \varrho(q),\ (c,q) \in \mathcal{M}_\odot[q_i]\,\} \quad \text{with} \quad q_i \in \mathcal{Q}^n, \tag{5.22}$$

called the sensorimotor obstruction neighborhood of the configuration q_i defined with respect to the motor neighborhood $\mathcal{M}_\odot[q_i]$. An alternative view is given by

$$\Lambda_{(\mathcal{B},\odot)}[q_i] = \{\,(c,\beta)\,|\,\beta = \varrho(q_i \bullet \tau_\blacktriangleright(c)),\ c \in C_\odot\,\} \quad \text{with} \quad q_i \in \mathcal{Q}^n, \tag{5.23}$$

which is entirely based on the combined mapping from C to $\mathcal{Q}^n$ given by Eq. (4.17) and the entity-specific mapping from $\mathcal{Q}^n$ to $\mathcal{B}$ given by Eq. (3.10). According to these definitions, it holds that

$$\underbrace{\mathcal{Q}^n \times (\overbrace{C \xrightarrow{\tau_*} C_* \xrightarrow{\tau_\cup} T_{id}\mathcal{Q}^n \xrightarrow{\exp} \mathcal{Q}^n}^{\tau_\blacktriangleright : C \to \mathcal{Q}^n}) \to \mathcal{Q}^n}_{\bullet\,:\,\mathcal{Q}^n \times \mathcal{Q}^n \to \mathcal{Q}^n} \xrightarrow{\varrho} \mathcal{B}. \tag{5.24}$$

In both cases the sensorimotor obstruction neighborhood $\Lambda_{(\mathcal{B},\odot)}[q_i]$ is a local reference frame where tuples $(c,\beta) \in \Lambda_{(\mathcal{B},\odot)}[q_i]$ represent information on empty and occupied configurations of the neighborhood of $q_i \in \mathcal{Q}^n$ by directly relating elements from $\beta \in \mathcal{B}$ to motor controls $c \in C_\odot$. A visualization of $\Lambda_{(\mathcal{B},\odot)}[q_i]$ is given in Fig. 5.4a.

We denote the tuple created from Eqs. (5.19) and (5.22) as the sensorimotor neighborhood of the configuration q_i given by

$$\Lambda_\odot[q_i] = (\,\Lambda_{(\mathcal{S},\odot)}[q_i], \Lambda_{(\mathcal{B},\odot)}[q_i]\,). \tag{5.25}$$

The sensorimotor neighborhood represents a local patch on $\mathcal{Q}^n$, where each tuple encodes the relation between a motor control and the sensory information that is perceived after its execution. As we did not establish any arbitrary coordinate transformation, both $\Lambda_{(\mathcal{S},\odot)}[q_i]$ and the sensorimotor obstruction neighborhood $\Lambda_{(\mathcal{B},\odot)}[q_i]$ represent said patch with respect to the agent-specific set of controls, so it is defined in terms of the agent's individual motor capabilities.

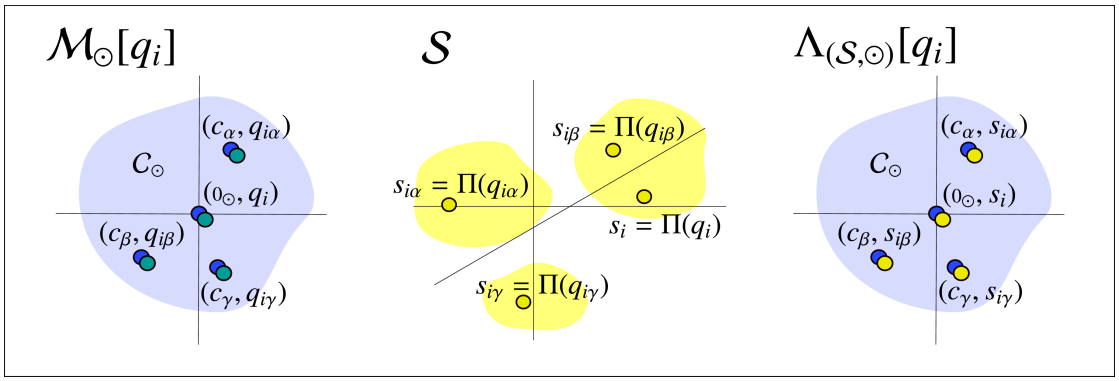
Another important property of both Eqs. (5.19) and (5.22) is that though the configuration q_i is required as a reference point on $\mathcal{Q}^n$, neither the tuples in $\Lambda_{(\mathcal{S},\odot)}[q_i]$ nor in $\Lambda_{(\mathcal{B},\odot)}[q_i]$ contain q_i or other elements from $\mathcal{Q}^n$. Thus, both sets are detached from $\mathcal{Q}^n$ as such they only require $C_\odot$, $\mathcal{S}$, and $\mathcal{B}$ to be defined.

We now combine the concept of the sensorimotor neighborhood with that of the sensorimotor chain by visualizing sensorimotor sample and obstruction neighborhoods being attached to every memory node ω_i. As a prerequisite, we have to recall that each memory node ω_i represents a distinct configuration $q_{(\omega,i)} \in \mathcal{Q}^n$ (see Eq. (5.12)). Though $q_{(\omega,i)} \in \mathcal{Q}^n$ is not directly observable, we can nevertheless define its neighborhoods by applying Eq. (5.20) which yields

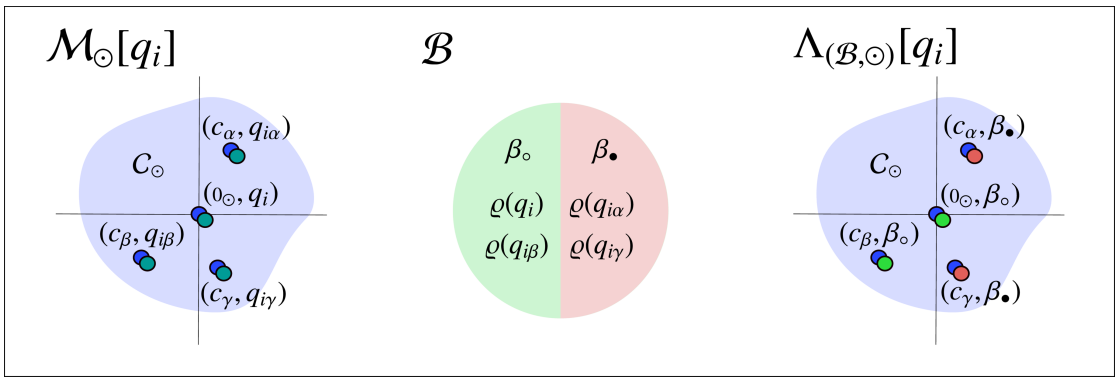
$$\Lambda_{(\mathcal{S},\odot)}[q_{(\omega,i)}] = \{\,(c,s)\,|\,s = \Pi(q_{(\omega,i)} \bullet \tau_\blacktriangleright(c)),\ c \in C_\odot\,\} \quad \text{with} \quad q_{(\omega,i)} \in \mathcal{Q}^n, \tag{5.26}$$

and Eq. (5.23), which yields

$$\Lambda_{(\mathcal{B},\odot)}[q_{(\omega,i)}] = \{\,(c,\beta)\,|\,\beta = \varrho(q_{(\omega,i)} \bullet \tau_\blacktriangleright(c)),\ c \in C_\odot\,\} \quad \text{with} \quad q_{(\omega,i)} \in \mathcal{Q}^n, \tag{5.27}$$



(a) Visualization of the sensorimotor sample neighborhood $\Lambda_{(\mathcal{S},\odot)}[q_i]$. Based on the motor neighborhood $\mathcal{M}_\odot[q_i]$, the sensorimotor sample neighborhood is constructed by applying the sample function Π to all configurations $q \in (c, q)$ with $(c, q) \in \mathcal{M}_\odot[q_i]$ and then associating the corresponding sample $\Pi(q)$ to every control c.



(b) Visualization of the sensorimotor obstruction neighborhood $\Lambda_{(\mathcal{B},\odot)}[q_i]$. Based on the motor neighborhood $\mathcal{M}_\odot[q_i]$, the sensorimotor obstruction neighborhood is constructed by applying the entity-specific occupancy function ρ to all configurations $q \in (c, q)$ with $(c, q) \in \mathcal{M}_\odot[q_i]$ and then associating the corresponding element from the boundary set to every control c.

Figure 5.4: Visualization of the sensorimotor sample neighborhood $\Lambda_{(\mathcal{S},\odot)}[q_i]$ and the sensorimotor obstruction neighborhood $\Lambda_{(\mathcal{B},\odot)}[q_i]$.

respectively. Both represent local reference frames with respect to the configuration $q_{(\omega,i)}$ and thus with respect to the memory node ω_i. More precisely, they define a local coordinate frame, given by $C_\odot \subseteq C \equiv \mathbb{R}^m$ (see Eqs. (4.3) and (4.8)).

While Eqs. (5.26) and (5.27) define the set of all possible samples and obstructions within the vicinity of a given memory node, we are more interested in the information we can extract from the sensorimotor chain explicitly. As given by Eq. (5.13) to each memory node ω_i the sample $s_{(\omega,i)} \in \mathcal{S}$ is associated. With respect to the notation of the sensorimotor sample neighborhood for the associated configuration $q_{(\omega,i)}$ given by Eq. (5.26) this translates to the tuple $(0_\odot, s_{(\omega,i)})$, where $0_\odot \in C_\odot$ is the zero control, expressing that no further transformation is needed to reach the configuration $q_{(\omega,i)}$, where the sample and $s_{(\omega,i)} \in \mathcal{S}$ has been perceived at. As it holds that $0_\odot \in C_\odot$ according to Eq. (5.26) this tuple naturally is an element of $\Lambda_{(\mathcal{S},\odot)}[q_{(\omega,i)}]$. The

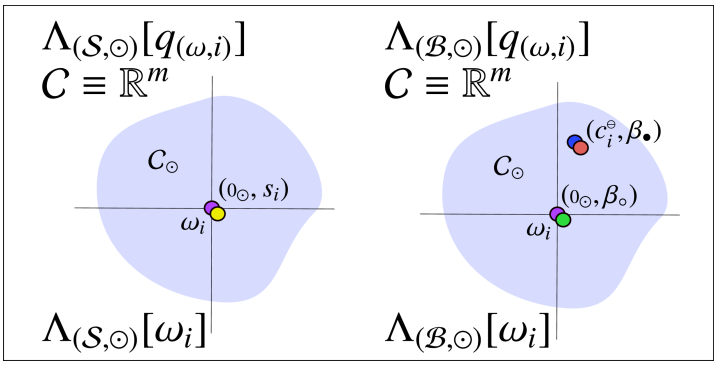


Figure 5.5: Example for sensorimotor sample and obstruction sets associated to the memory node ω_i. The sensorimotor sample set $\Lambda_{(\mathcal{S},\odot)}[\omega_i]$ contains only single tuple $(0_\odot, s_i)$, where $s_i = \Pi(q_{(\omega,i)})$. The sensorimotor obstruction set $\Lambda_{(\mathcal{B},\odot)}[\omega_i]$ contains two tuples $(0_\odot, \beta_\circ)$ and $(c_i^\ominus, \beta_\bullet)$. The former denotes that the configuration $q_{(\omega,i)}$ is unobstructed, i.e., that it holds that $\varrho(q_{(\omega,i)}) = \beta_\circ$. The latter denotes that the scheduled control from ω_i to ω_{i+1} has only been executed partially due to an obstruction in $\mathcal{Q}^n$. It thus represents that $c_i^\ominus$ can not be executed, which correspond to the associated $\beta_\bullet \in \mathcal{B}$.

sensorimotor sample set of the memory node ω_i is therefore given by

$$\Lambda_{(\mathcal{S},\odot)}[\omega_i] = \{\, (0_\odot, s_i) \,\}, \tag{5.28}$$

where it holds that $\Lambda_{(\mathcal{S},\odot)}[\omega_i] \subseteq \Lambda_{(\mathcal{S},\odot)}[q_{(\omega,i)}]$.

We have previously established that each memory node corresponds to a distinct configuration $q_{(\omega,i)} \in \mathcal{Q}^n$ the entity either traversed through or currently is at. Naturally for all configurations $q_{(\omega,i)}$ associated to the memory nodes ω_i from Ω_t it holds that $\varrho(q_{(\omega,i)}) = \beta_\circ$, i.e., the configuration associated to a given memory node ω_i is unobstructed. With respect to the notation of the sensorimotor obstruction neighborhood for the associated configuration $q_{(\omega,i)}$ given by Eq. (5.27) this translates to the tuple $(0_\odot, \beta_\circ)$, where $0_\odot \in C_\odot$ is the zero control, expressing that no further transformation is needed to reach the unobstructed $q_{(\omega,i)}$. As it holds that $0_\odot \in C_\odot$ according to Eq. (5.27) this tuple naturally is an element of $\Lambda_{(\mathcal{B},\odot)}[q_{(\omega,i)}]$. In cases, where $c_i^\ominus \neq c_i^\otimes$ the former encodes an obstruction in $\mathcal{Q}^n$ relative to $q_{(\omega,i)}$. However, in this case we use the notion of occupancy in a slightly different way: rather than expressing that the configuration $q_{(\omega,i)}^\ominus = q_{(\omega,i)} \bullet \tau_\blacktriangleright(c_i^\ominus)$ is occupied (which we do not know, as given in Section 3.3.1, Eqs. (3.25) to (3.26)), we utilize it to indicate that the path from $q_{(\omega,i)}$ to $q_{(\omega,i)}^\ominus$, given by $c_i^\ominus$, is obstructed. This translates to the tuple $(c_i^\ominus, \beta_\bullet)$, expressing that $c^\ominus$ was not executable at q_i. As it holds that $c_i^\ominus \in C_\odot$, Eq. (5.27) can be applied here as well, thus the tuple is an element of $\Lambda_{(\mathcal{B},\odot)}[q_{(\omega,i)}]$ too. The sensorimotor obstruction set of the memory node ω_i is therefore given as

$$\Lambda_{(\mathcal{B},\odot)}[\omega_i] = \begin{cases} \{\, (0_\odot, \beta_\circ),\ (c_i^\ominus, \beta_\bullet) \,\}, & \text{if} \quad c_i^\ominus \neq c_i^\otimes \\ \{\, (0_\odot, \beta_\circ) \,\}, & \text{if} \quad c_i^\ominus = c_i^\otimes \quad \text{or} \quad i = t, \end{cases} \tag{5.29}$$

where it holds that $\Lambda_{(\mathcal{B},\odot)}[\omega_i] \subseteq \Lambda_{(\mathcal{B},\odot)}[q_{(\omega,i)}]$. Both the sensorimotor sample set $\Lambda_{(\mathcal{S},\odot)}[\omega_i]$ and the sensorimotor obstruction set $\Lambda_{(\mathcal{B},\odot)}[\omega_i]$ are visualized in Fig. 5.5.

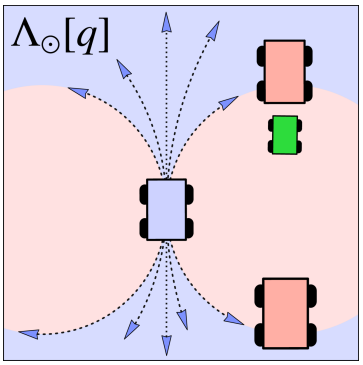


Figure 5.6: Visualization of the sensorimotor neighborhood of a nonholonomic robot using a slip/skid steering configuration. Blue areas correspond to configurations that are reachable by executing a single control and are represented within the sensorimotor neighborhood. Red areas, correspond to configurations the system can reach only by executing multiple, non-commutative controls, e.g. parallel parking, and are not represented within the sensorimotor neighborhood. As a consequence, parts of the obstructions (red robots) as well as the parking space (green robot) are not represented within the sensorimotor neighborhood.

We denote the tuple created from Eqs. (5.28) and (5.29), the sensorimotor set of the memory node ω_i given as

$$\Lambda_{\odot}[\omega_i] = (\Lambda_{(\mathcal{S},\odot)}[\omega_i], \Lambda_{(\mathcal{B},\odot)}[\omega_i]). \tag{5.30}$$

5.2.2.2 Extended Sensorimotor Neighborhoods

A drawback of the established definition of the sensorimotor neighborhood is that its local dimension is fixed to that of the control space $C_{\odot} \subseteq C \equiv \mathbb{R}^m$. This means that the local neighborhood only accounts for configurations that are reachable by executing a single control. For holonomic systems this is a valid approach, as the dimension of the control and the configuration space match (see Section 3.7). However, for nonholonomic systems this might be a suboptimal.

If a given reference frame at $q \in Q^n$ only represent configurations the agent can reach directly, there will be configurations in Q^n that are, with respect to the structure of Q^n, close to q, but can not be expressed as elements of the sensorimotor neighborhood. As an example we revisit the parallel-parking scenario from Section 3.7, Fig. 3.9b, as given in Fig. 5.6. As established in Section 3.7, a robot using a slip/skid steering configuration is a nonholonomic system. Thus, there exist configurations $q \in Q^n$ such a system can not reach when utilizing only a single control. The areas that are colored in blue correspond to configurations that are reachable by executing a single control and thus configurations that are represented within the sensorimotor neighborhood. The red areas, however, are configurations the system can reach only by executing multiple, non-commutative controls, e.g. by performing parallel parking. Even though the parking space (depicted as a green robot), is not contained within the sensorimotor neighborhood, information on its relative location might be relevant for planning the parking maneuver. The same holds for the obstructions (red robots), that are only partially represented within the

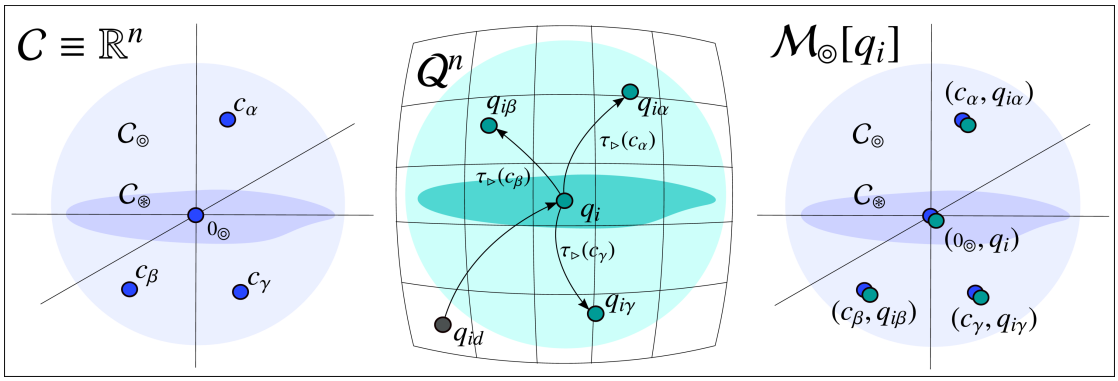


Figure 5.7: Visualization of the extended motor neighborhood $\mathcal{M}_\circledcirc[q_i]$. Applying $\tau_\triangleright$ to elements $0_\circledcirc$, c_α, c_β, c_γ from $C_\circledcirc$ yields the corresponding group elements $q_{id}, q_\alpha, q_\beta, q_\gamma$. Applying these to q_i yields q_{ij} as $q_i \bullet \tau_\triangleright(c_j)$. Naturally, q_i is given as $q_i = q_{id} \bullet q_i$. Thus, $\mathcal{M}_\circledcirc[q_i]$ is the set of all tuples (c, q), where c is an element from the extended control space $C_\circledcirc$ and q is the associated configuration from $\mathcal{Q}^n$, created by applying $\tau_\triangleright(c)$ to the origin of the extended motor neighborhood q_i.

sensorimotor neighborhood. To account for nonholonomic scenarios we extend the sensorimotor neighborhoods to not only represent a reachable subset on $\mathcal{Q}^n$ but to correspond to a larger neighborhood on $\mathcal{Q}^n$.

As a prerequisite we define an extended sensorimotor neighborhood with respect to a different motor cover which is based on the envelope $C_\circledcirc$ (see Eq. (4.10)). As given by Eq. (4.14), the function $\tau_\triangleright$ allows us to map elements from the extended control space C_* to the configuration space $\mathcal{Q}^n$. As it holds $C_\circledcirc \subseteq C_*$ the mapping $\tau_\triangleright$ can naturally be applied to elements $c \in C_\circledcirc$ as well. Mapping elements from C_* to $\mathcal{Q}^n$ is performed by applying the subsequent transformations the function $\tau_\triangleright$ is composed of, i.e., $\tau_\cup$ (Eq. (4.12)) and the exponential map (Eq. (3.62)). Thus, it holds that

$$\overbrace{C_* \xrightarrow{\tau_\cup} T_{id}\mathcal{Q}^n \xrightarrow{\exp} \mathcal{Q}^n}^{\tau_\triangleright : C_* \to \mathcal{Q}^n}. \tag{5.31}$$

Similarly to Eq. (5.16), we denote $\mathcal{R}_\circledcirc$ the extended motor cover of $C_\circledcirc$ which is the set of tuples of all elements in $C_\circledcirc$ and their corresponding transformations on $\mathcal{Q}^n$ given as

$$\mathcal{R}_\circledcirc = \{\, (c, q) \,|\, q = \tau_\triangleright(c),\ c \in C_\circledcirc \,\}. \tag{5.32}$$

Applying the extended motor cover $\mathcal{R}_\circledcirc$ to an arbitrary configuration $q_i \in \mathcal{Q}^n$ yields

$$\mathcal{M}_\circledcirc[q_i] = \{\, (c, q) \,|\, q = q_i q_j,\ (c, q_j) \in \mathcal{R}_\circledcirc \,\} \quad \text{with} \quad q_i \in \mathcal{Q}^n, \tag{5.33}$$

i.e.,

$$\mathcal{M}_\circledcirc[q_i] = \{\, (c, q) \,|\, q = q_i \tau_\triangleright(c),\ c \in C_\circledcirc \,\} \quad \text{with} \quad q_i \in \mathcal{Q}^n, \tag{5.34}$$

the extended motor neighborhood of the configuration q_i which is visualized in Figure 5.7.

To incorporate information on samples in $\mathcal{M}_{\circledcirc}[q_i]$ we apply Eq. (4.52) which yields

$$\Lambda_{(\mathcal{S},\circledcirc)}[q_i] = \{\,(c,s)\,|\,s = \Pi(q),\,(c,q) \in \mathcal{M}_{\circledcirc}[q_i]\,\} \quad \text{with} \quad q_i \in Q^n, \tag{5.35}$$

as the extended sensorimotor sample neighborhood of the configuration q_i defined with respect to $\mathcal{M}_{\odot}[q_i]$. Consequently

$$\Lambda_{(\mathcal{S},\circledcirc)}[q_i] = \{\,(c,s)\,|\,s = \Pi(q_i \bullet \tau_{\triangleright}(c)),\,c \in C_{\circledcirc}\,\} \quad \text{with} \quad q_i \in Q^n \tag{5.36}$$

utilizes the composition of Eqs. (3.44), (4.14) and (4.52) which yields

$$\underbrace{Q^n \times (\overbrace{C_* \xrightarrow{\tau_{\cup}} T_{id}Q^n \xrightarrow{\exp} Q^n}^{\tau_{\triangleright}\,:\,C_* \to Q^n}) \to Q^n}_{\bullet\,:\,Q^n \times Q^n \to Q^n} \xrightarrow{\Pi} \mathcal{S}. \tag{5.37}$$

To incorporate obstructions in $\mathcal{M}_{\circledcirc}[q_i]$ we apply Eq. (3.10) which yields

$$\Lambda_{(\mathcal{B},\circledcirc)}[q_i] = \{\,(c,\beta)\,|\,\beta = \varrho(q),\,(c,q) \in \mathcal{M}_{\circledcirc}[q_i]\,\} \quad \text{with} \quad q_i \in Q^n, \tag{5.38}$$

the extended sensorimotor obstruction neighborhood of the configuration q_i defined with respect to $\mathcal{M}_{\odot}[q_i]$. However

$$\Lambda_{(\mathcal{B},\circledcirc)}[q_i] = \{\,(c,\beta)\,|\,\beta = \varrho(q_i\tau_{\triangleright}(c)),\,c \in C_{\circledcirc}\,\} \quad \text{with} \quad q_i \in Q^n \tag{5.39}$$

utilizes the composition of Eqs. (3.10), (3.44) and (4.14) as

$$\underbrace{Q^n \times (\overbrace{C_* \xrightarrow{\tau_{\cup}} T_{id}Q^n \xrightarrow{\exp} Q^n}^{\tau_{\triangleright}\,:\,C_* \to Q^n}) \to Q^n}_{\bullet\,:\,Q^n \times Q^n \to Q^n} \xrightarrow{\varrho} \mathcal{B}. \tag{5.40}$$

We denote the tuple created from Eqs. (5.35) and (5.38) given as

$$\Lambda_{\circledcirc}[q_i] = (\,\Lambda_{(\mathcal{S},\circledcirc)}[q_i], \Lambda_{(\mathcal{B},\circledcirc)}[q_i]\,). \tag{5.41}$$

the extended sensorimotor neighborhood of the configuration q_i. Both $\Lambda_{(\mathcal{S},\circledcirc)}[q_i]$ and $\Lambda_{(\mathcal{B},\circledcirc)}[q_i]$ are visualized in Fig. 5.8.

Similar to Eqs. (5.26) and (5.27) we can associate the arbitrary configurations $q_i \in Q^n$ to the distinct configurations associated to an arbitrary memory node ω_i, i.e., attach the extended sensorimotor sample and obstruction neighborhoods to $q_{(\omega,i)}$ which yields

$$\Lambda_{(\mathcal{S},\circledcirc)}[q_{(\omega,i)}] = \{\,(c,s)\,|\,s = \Pi(q_{(\omega,i)} \bullet \tau_{\triangleright}(c)),\,c \in C_{\circledcirc}\,\} \tag{5.42}$$

and

$$\Lambda_{(\mathcal{B},\circledcirc)}[q_{(\omega,i)}] = \{\,(c,\beta)\,|\,\beta = \varrho(q_{(\omega,i)} \bullet \tau_{\triangleright}(c)),\,c \in C_{\circledcirc}\,\}. \tag{5.43}$$

To establish the relation between $\Lambda_{(\mathcal{S},\circledcirc)}[q_{(\omega,i)}]$ and $\Lambda_{(\mathcal{B},\circledcirc)}[q_{(\omega,i)}]$ and the sensorimotor sample and obstruction sets we utilize τ_* given by Eq. (4.5). Applying τ_* to every control within the

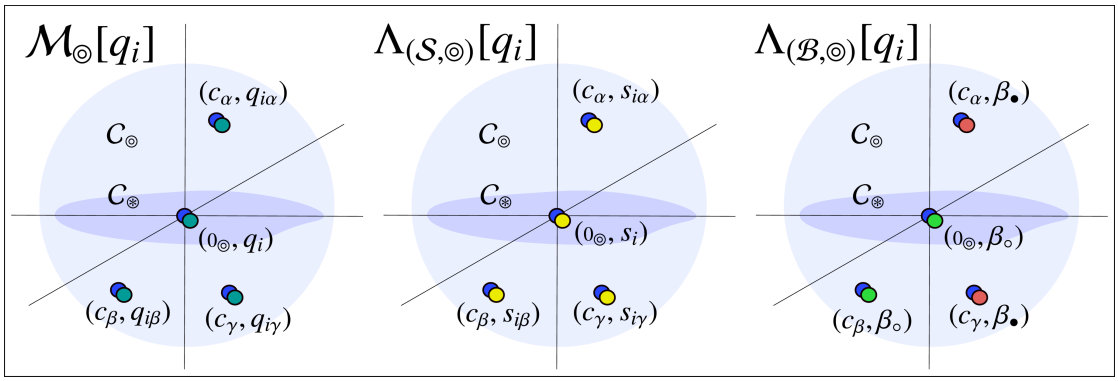


Figure 5.8: Visualization of the extended sensorimotor sample neighborhood $\Lambda_{(\mathcal{S},\circledcirc)}[q_i]$ and the extended sensorimotor obstruction neighborhood $\Lambda_{(\mathcal{B},\circledcirc)}[q_i]$. The former is constructed by applying the sample function Π to all configurations $q \in (c, q)$ with $(c, q) \in \mathcal{M}_{\circledcirc}[q_i]$ and then associating to every control c the corresponding sample $\Pi(q)$. Correspondingly, the latter is built by applying the entity-specific occupancy function ϱ to all configurations $q \in (c, q)$ with $(c, q) \in \mathcal{M}_{\circledcirc}[q_i]$.

sensorimotor sample set Eq. (5.28) yields the extended sensorimotor sample set of the memory node ω_i given as

$$\Lambda_{(\mathcal{S},\circledcirc)}[\omega_i] = \{\, (\tau_*(0_\odot), s_i) \,\}. \tag{5.44}$$

Accordingly we apply τ_* to every control within the sensorimotor obstruction set Eq. (5.29) which yields the extended sensorimotor obstruction set of the memory node ω_i given as

$$\Lambda_{(\mathcal{B},\circledcirc)}[\omega_i] = \begin{cases} \{\, (\tau_*(0_\odot), \beta_\circ), (\tau_*(c_i^{\ominus}), \beta_\bullet) \,\}, & \text{if} \quad c_i^{\ominus} \neq c_i^{\otimes} \\ \{\, (\tau_*(0_\odot), \beta_\circ) \,\}, & \text{if} \quad c_i^{\ominus} = c_i^{\otimes} \quad \text{or} \quad i = t, \end{cases} \tag{5.45}$$

In both cases it holds that $\tau_*(0_\odot) = 0_{\circledast}$ as the zero control from $C_\odot$ is mapped to the zero control in $C_{\circledast}$, where naturally $0_{\circledast} = 0_{\circledcirc}$. We denote the tuple created from Eqs. (5.44) and (5.45) the extended sensorimotor set of the memory node ω_i with

$$\Lambda_{\circledcirc}[\omega_i] = (\, \Lambda_{(\mathcal{S},\circledcirc)}[\omega_i], \Lambda_{(\mathcal{B},\circledcirc)}[\omega_i] \,) \tag{5.46}$$

which is visualized in Fig. 5.9. Controls c from tuples $(c, s) \in \Lambda_{(\mathcal{S},\circledcirc)}[q_{(\omega,i)}]$ and $(c, \beta) \in \Lambda_{(\mathcal{B},\circledcirc)}[q_{(\omega,i)}]$ are from $C_{\circledcirc}$, where controls c from $(c, s) \in \Lambda_{(\mathcal{S},\circledcirc)}[\omega_i]$ and $(c, \beta) \in \Lambda_{(\mathcal{B},\circledcirc)}[\omega_i]$ are from $C_{\circledast}$. As $C_{\circledast} \subseteq C_{\circledcirc}$ (see Eq. (4.9)) it follows that

$$\Lambda_{(\mathcal{S},\circledcirc)}[\omega_i] \subseteq \Lambda_{(\mathcal{S},\circledcirc)}[q_{(\omega,i)}] \tag{5.47}$$

and accordingly

$$\Lambda_{(\mathcal{B},\circledcirc)}[\omega_i] \subseteq \Lambda_{(\mathcal{B},\circledcirc)}[q_{(\omega,i)}]. \tag{5.48}$$

Thus, the extended sensorimotor set of ω_i is a subset of the extended sensorimotor neighborhood of its corresponding configuration $q_{(\omega,i)}$. This relation allows us to utilize extended sensorimotor neighborhoods $\Lambda_{\circledcirc}[q_{(\omega,i)}]$ as local reference frames which ultimately will serve as building blocks for the sensorimotor map.

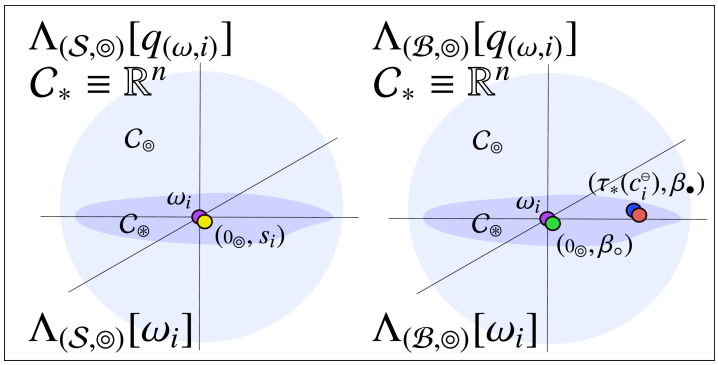


Figure 5.9: Example for extended sensorimotor sample and obstruction sets associated to the memory node ω_i. The former contains only the single tuple $(0_\circledcirc, s_i)$, where $s_i = \Pi(q_{(\omega,i)})$. The latter contains two tuples $(0_\circledcirc, \beta_\circ)$ and $(\tau_*(c_i^\ominus), \beta_\bullet)$ The first one denotes that the configuration $q_{(\omega,i)}$ is unobstructed. The second one denotes that the scheduled control from ω_i to ω_{i+1} has only been executed partially due to an obstruction in $\mathcal{Q}^n$. Notably tuples from $\Lambda_{(\mathcal{S},\circledcirc)}[\omega_i]$ or $\Lambda_{(\mathcal{B},\circledcirc)}[\omega_i]$ are either associated to the origin of $C_\circledcirc$ or lie within $C_\circledast \in C_*$.

5.2.3 Reference Frame Transformations

The established extended sensorimotor neighborhoods allow us to represent a local patch around a given $q \in \mathcal{Q}^n$. The next step is to formally define how to relate two given extended sensorimotor neighborhoods to each other which ultimately will allow us to relate arbitrary extended sensorimotor sets to each other. This is required as we aim for developing a consistent sensorimotor representation of $\mathcal{Q}^n$ which, accordingly, has to account for the transitions between individual patches. These transitions can be expressed with respect to the group operations.

5.2.3.1 Transformations on the Configuration Space

As a first step we look at the topology of the configuration space $\mathcal{Q}^n$. We established that $\mathcal{Q}^n$ corresponds to a matrix Lie group (see Section 3.5), i.e., a smooth manifold which has a group structure. According to the group axioms (Eq. (3.45)) in conjunction with the group operation (Eq. (3.44)) it therefore holds that for any two given group elements $q_i, q_j \in \mathcal{Q}^n$ that the transformation from q_i to q_j is given as

$$q_{ij} = q_i^{-1} \bullet q_j \tag{5.49}$$

and the transformation from q_j to q_i respectively as

$$q_{ji} = q_j^{-1} \bullet q_i, \tag{5.50}$$

where naturally it holds that

$$q_{ij} \bullet q_{ji} = q_{ji} \bullet q_{ij} = id. \tag{5.51}$$

Since we defined sensorimotor neighborhoods as a reference frame with respect to an origin, given as a configuration $q \in \mathcal{Q}^n$, we can utilize these transformations to relate sensorimotor neighborhoods to each other. Consider the extended sensorimotor neighborhoods $\Lambda_\circledcirc[q_i]$ and

$\Lambda_{\circledcirc}[q_j]$ associated to the configurations $q_i, q_j \in \mathcal{Q}^n$, respectively. The transformation from the origin of $\Lambda_{\circledcirc}[q_i]$ to the origin of $\Lambda_{\circledcirc}[q_j]$ is given by Eq. (5.49) and correspondingly the transformation from the origin of $\Lambda_{\circledcirc}[q_j]$ to the origin of $\Lambda_{\circledcirc}[q_i]$ is given by Eq. (5.50). As we utilize matrix Lie groups that are path-connected (see Eq. (3.20)), these transformations are guaranteed to exist. To calculate the control $c \in C_*$ that corresponds to these transformations, we apply $\tau_{\blacktriangleleft}$ (see Eq. (4.40)). Thus, the control corresponding to the transformation from the origin of $\Lambda_{\circledcirc}[q_i]$ to the origin of $\Lambda_{\circledcirc}[q_j]$ is given by

$$c_{ij} = \tau_{\blacktriangleleft}(q_i^{-1} q_j) \tag{5.52}$$

and accordingly the control corresponding to the transformation from the origin of $\Lambda_{\circledcirc}[q_j]$ to the origin of $\Lambda_{\circledcirc}[q_i]$ corresponds to

$$c_{ji} = \tau_{\blacktriangleleft}(q_j^{-1} q_i), \tag{5.53}$$

where $c_{ij}, c_{ji} \in C_*$ and it naturally holds that $c_{ij} = -c_{ji}$ and $c_{ji} = -c_{ij}$. As the calculation utilizes the matrix logarithm it suffers from the problems depicted in Section 4.3.3.2 with respect to the result potentially not being uniquee. However, even if $c = \tau_{\blacktriangleleft}(q)$ for a $q \in \mathcal{Q}^n$ is not unique, it is guaranteed that there exist at least one $c \in C_*$ for which $\tau_{\triangleright}(c) = q$ holds. This is due to the connection between the Lie algebra and the Lie group that implies that for every group element there exists at least one Lie algebra element, i.e., a tangent vector from $T_{id}\mathcal{Q}^n$ that, applying the exponential map to it, creates said group element. This means that even if calculating $\tau_{\blacktriangleleft}(q_i^{-1} q_j)$ or $\tau_{\blacktriangleleft}(q_j^{-1} q_i)$ from the configuration is not necessarily possible (as the result would be not unique) the corresponding controls $c_{ij}, c_{ji} \in C_*$ exist anyway. Though c_{ij} and c_{ji} are per definition elements of C_* it does not necessarily hold that they lie within the Euclidean n-ball given by $C_{\circledcirc}$, i.e., $c_{ij}, c_{ji} \in C_{\circledcirc}$ is not necessarily true. This is due to the fact that, even though the transformation between the two neighborhoods is guaranteed to exist, they may be too far away from each other to be connected by a single transformation, i.e., a single control $c \in C_{\circledcirc}$. To represent these cases we denote two extended sensorimotor neighborhoods associated to $q_i, q_j \in \mathcal{Q}^n$ respectively to be connected with

$$\text{connected}_{\circledcirc} : \Lambda_{\circledcirc}[q_i] \times \Lambda_{\circledcirc}[q_j] \rightarrow \{true, false\} \tag{5.54}$$

which is given by

$$\text{connected}_{\circledcirc}(\,\Lambda_{\circledcirc}[q_i],\, \Lambda_{\circledcirc}[q_j]\,) = \begin{cases} true, & \text{if} \quad \exists\, c \in C_{\circledcirc} : \tau_{\triangleright}(c) = q_{ij} \\ false, & \text{otherwise} \end{cases} \tag{5.55}$$

As $C_{\circledcirc}$ is defined as an Euclidean n-ball in C_*, two extended neighborhoods are connected if the images of their envelopes $C_{\circledcirc}$ under $\tau_{\triangleright}$ overlap in $\mathcal{Q}^n$ in such a way that their respective origins are contained in both images. As $C_{\circledcirc}$ is defined with respect to the radius $\epsilon_{\circledcirc}$ (see Section 4.3.2), Eq. (5.54) depends on this parameter in such a way that increasing the radius $\epsilon_{\circledcirc}$ corresponds to increasing the set of potentially connected extended sensorimotor neighborhoods on $\mathcal{Q}^n$. As a limit case it holds that for $\epsilon_{\circledcirc} = \infty$ the envelope $C_{\circledcirc}$ would match C_*. In this case the image of $(C_{\circledcirc} \equiv C_*)$ under $\tau_{\circlearrowleft}$ (see Eq. (4.12)) would be equivalent to $T_{id}\mathcal{Q}^n$. Thus, the image of

($C_{\circledcirc} \equiv C_*$) under $\tau_{\triangleright}$ (see Eq. (4.14)) would correspond to $\mathcal{Q}^n$. This ultimately states the obvious that if $\epsilon_{\circledcirc} = \infty$ it holds that

$$\forall q_i, q_j \in \mathcal{Q}^n : \exists c \in C_* : \tau_{\triangleright}(c) = q_{ij} \quad \text{with} \quad q_{ij} = q_i^{-1} \bullet q_j \tag{5.56}$$

which corresponds to the definition of $\mathcal{Q}^n$ being path-connected (see Eq. (3.20) and thus it holds that

$$\forall q_i, q_j \in \mathcal{Q}^n : \text{connected}_{\circledcirc}(\, \Lambda_{\circledcirc}[q_i],\, \Lambda_{\circledcirc}[q_j]\,) = \mathit{true} \quad \text{iff} \quad \epsilon_{\circledcirc} = \infty, \tag{5.57}$$

i.e., that for $\epsilon_{\circledcirc} = \infty$ arbitrary extended sensorimotor neighborhoods are connected. Notably, however, we can not make any statement on whether the path between q_i and q_j is free, as being path-connected is a mathematical property of the matrix Lie group, while a path being unobstructed or obstructed depends on the environment and the agent with respect to ϱ (see Eq. (3.10)).

In addition, we can not make any assumption whether either c_{ij} or c_{ji} are valid controls the agent can execute. As an example we consider two connected extended sensorimotor neighborhoods, with $c_{ij}, c_{ji} \in C_{\circledcirc}$. However, it does not necessarily hold that either c_{ij} or c_{ji} are in $C_{\circledast}$. This holds, as $C_{\circledast}$, the image of $C_{\odot}$ under τ_*, is only a subset of $C_{\circledcirc}$. This is especially severe in the nonholonomic case, where elements from $C_{\circledast}$ corresponds to subset of $C_{\circledcirc}$. To account for these situations we denote two sensorimotor neighborhoods to be directly connected with

$$\text{connected}_{\odot} : \Lambda_{\circledcirc}[q_i] \times \Lambda_{\circledcirc}[q_j] \to \{\mathit{true}, \mathit{false}\} \tag{5.58}$$

which is given by

$$\text{connected}_{\odot}(\, \Lambda_{\circledcirc}[q_i],\, \Lambda_{\circledcirc}[q_j]\,) = \begin{cases} \mathit{true}, & \text{if} \quad \exists\, c_{ij} \in C_{\odot} : \tau_{\blacktriangleright}(c_{ij}) = q_{ij} \\ \mathit{false}, & \text{otherwise.} \end{cases} \tag{5.59}$$

This means that two extended sensorimotor neighborhoods are directly connected if for the transform from q_i to q_j there exists a corresponding element from $C_{\odot}$. However, from $C_{\odot}$ being symmetric (see Section 4.2) follows that, if the control $c_{ij} \in C_{\odot}$ exists that describes the transform from q_i to q_j, the inverse control $c_{ji} = -c_{ij} \in C_{\odot}$ exists too and corresponds to the transform from q_j to q_i.

5.2.3.2 Transformations between Neighborhoods

Being able to relate the origins of two extended sensorimotor neighborhoods to each other allows us to perform coordinate transformations, where the aim is to represent elements from one extended sensorimotor neighborhood with respect to the other one. Let $\Lambda_{\circledcirc}[q_i]$ and $\Lambda_{\circledcirc}[q_j]$ be the two connected extended sensorimotor neighborhoods at the configurations $q_i, q_j \in \mathcal{Q}^n$ respectively. The transformation from the origin of $\Lambda_{\circledcirc}[q_i]$ to the origin of $\Lambda_{\circledcirc}[q_j]$ is given as $q_{ij} = q_i^{-1} q_j$. For the corresponding control c_{ij} it holds per definition that $c_{ij} = \tau_{\blacktriangleleft}(q_{ij})$ and $q_{ij} = \tau_{\triangleright}(c_{ij})$. Looking at the extended sensorimotor neighborhood $\Lambda_{\circledcirc}[q_j]$ it holds that tuples $(c_{jk}, s_k) \in \Lambda_{(\mathcal{S},\circledcirc)}[q_j]$ and $(c_{jk}, \beta_k) \in \Lambda_{(\mathcal{B},\circledcirc)}[q_j]$ correspond to the configuration q_k in the vicinity

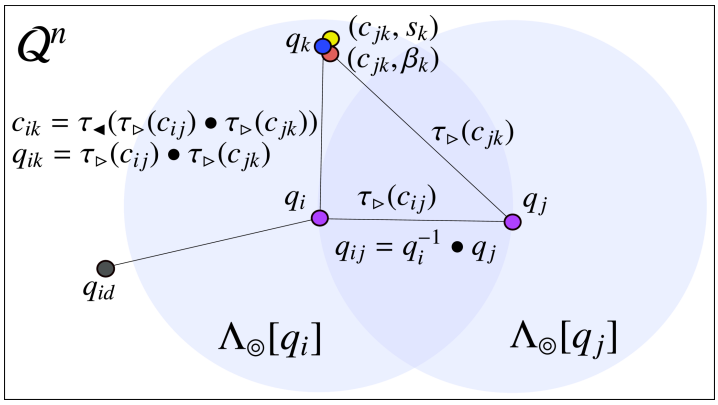


Figure 5.10: Visualization of the reference frame transformation between the two directly connected extended sensorimotor neighborhoods $\Lambda_\circledcirc[q_i]$ and $\Lambda_\circledcirc[q_j]$. The transformation from the origin of $\Lambda_\circledcirc[q_i]$ to the origin of $\Lambda_\circledcirc[q_j]$ is given as $q_{ij} = q_i^{-1} \bullet q_j$ which corresponds to $\tau_\triangleright(c_{ij})$. The transformation from q_j to arbitrary tuples (c_{jk}, s_k) and (c_{jk}, β_k) from $\Lambda_{(\mathcal{S},\circledcirc)}[q_j]$ and $\Lambda_{(\mathcal{B},\circledcirc)}[q_j]$, respectively, is given as $\tau_\triangleright(c_{jk})$. The transformation from q_i to q_k is thus given as $q_{ik} = \tau_\triangleright(c_{ij}) \bullet \tau_\triangleright(c_{jk})$. The matching control naturally is $c_{ik} = \tau_\blacktriangleleft(\tau_\triangleright(c_{ij}) \bullet \tau_\triangleright(c_{jk}))$.

of q_j, where $s_k \in \mathcal{S}$ can be perceived and $\beta_k \in \mathcal{B}$ is valid, respectively. The control $c_{jk} \in C_\circledcirc$ encodes the transformation from q_j to q_k as it holds that $q_{jk} = \tau_\triangleright(c_{jk})$. Combining the transformation from the origin of $\Lambda_\circledcirc[q_i]$ to the origin of $\Lambda_\circledcirc[q_j]$ and the transformation from the origin of $\Lambda_\circledcirc[q_j]$ to the configuration q_k yields

$$q_{ik} = q_{ij} \bullet q_{jk} = \tau_\triangleright(c_{ij}) \bullet \tau_\triangleright(c_{jk}) \tag{5.60}$$

with $q_{ik} \in Q^n$. The corresponding control can be calculated by applying Eq. (4.38) which yields

$$c_{ik} = \tau_\blacktriangleleft(q_{ik}) = \tau_\blacktriangleleft(q_{ij} \bullet q_{jk}). = \tau_\blacktriangleleft(\tau_\triangleright(c_{ij}) \bullet \tau_\triangleright(c_{jk})). \tag{5.61}$$

The control c_{ik} encodes the transformation that has to be performed to traverse from q_i to q_k, a visualization of which is given in Fig. 5.10.

Applying Eq. (5.61) to all tuples from $\Lambda_{(\mathcal{S},\circledcirc)}[q_j]$ yields

$$\Lambda_{(\mathcal{S},\circledcirc)}[q_i \triangleleft q_j] = \{\, (c, s) \,|\, c = \tau_\blacktriangleleft(\tau_\triangleright(c_{ij}) \bullet \tau_\triangleright(c_{jk})), (c_{jk}, s) \in \Lambda_{(\mathcal{S},\circledcirc)}[q_j],\ c \in C_\circledcirc \,\} \tag{5.62}$$

and correspondingly for tuples from $\Lambda_{(\mathcal{B},\circledcirc)}[q_j]$

$$\Lambda_{(\mathcal{B},\circledcirc)}[q_i \triangleleft q_j] = \{\, (c, \beta) \,|\, c = \tau_\blacktriangleleft(\tau_\triangleright(c_{ij}) \bullet \tau_\triangleright(c_{jk})), (c_{jk}, \beta) \in \Lambda_{(\mathcal{B},\circledcirc)}[q_j],\ c \in C_\circledcirc \,\}, \tag{5.63}$$

where we explicitly restrict the tuples to those who are within the neighborhood of q_i, i.e., for which holds that $c \in C_\circledcirc$. Naturally if it holds that $q_i = q_j$ the trivial case is given as

$$\Lambda_{(\mathcal{S},\circledcirc)}[q_i \triangleleft q_j] = \Lambda_{(\mathcal{S},\circledcirc)}[q_i] = \Lambda_{(\mathcal{S},\circledcirc)}[q_j] \tag{5.64}$$

and correspondingly

$$\Lambda_{(\mathcal{B},\circledcirc)}[q_i \triangleleft q_j] = \Lambda_{(\mathcal{B},\circledcirc)}[q_i] = \Lambda_{(\mathcal{B},\circledcirc)}[q_j]. \tag{5.65}$$

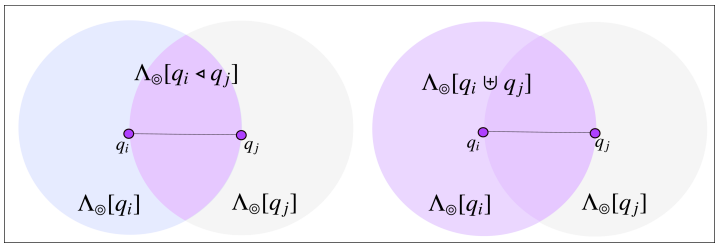


Figure 5.11: Difference between the reference frame transformation $\Lambda_{\circledcirc}[q_i \triangleleft q_j]$ and the projection set $\Lambda_{\circledcirc}[q_i \uplus q_j]$. The former corresponds to tuples from $\Lambda_{(\mathcal{B},\circledcirc)}[q_j]$ that are represented with respect to $\Lambda_{(\mathcal{B},\circledcirc)}[q_i]$ and lie within the Euclidean n-ball $C_{\circledcirc}$ centered on q_i. The latter corresponds to the union of tuples from $\Lambda_{\circledcirc}[q_i]$ and $\Lambda_{\circledcirc}[q_i \triangleleft q_j]$.

We denote the tuple created from Eqs. (5.62) and (5.63) the reference frame transformation from the extended sensorimotor neighborhood $\Lambda_{\circledcirc}[q_j]$ to $\Lambda_{\circledcirc}[q_i]$ given as

$$\Lambda_{\circledcirc}[q_i \triangleleft q_j] = (\, \Lambda_{(\mathcal{S},\circledcirc)}[q_i \triangleleft q_j], \Lambda_{(\mathcal{B},\circledcirc)}[q_i \triangleleft q_j] \,). \tag{5.66}$$

Equations (5.62) and (5.63) effectively allow us to transform tuples from one extended neighborhood to another, i.e., represent elements from one neighborhood with respect to the reference frame of another one. To account for both the transformed elements from $\Lambda_{\circledcirc}[q_j]$ and the elements originally contained in $\Lambda_{\circledcirc}[q_i]$ we define

$$\Lambda_{(\mathcal{S},\circledcirc)}[q_i \uplus q_j] = \Lambda_{(\mathcal{S},\circledcirc)}[q_i] \bigcup \Lambda_{(\mathcal{S},\circledcirc)}[q_i \triangleleft q_j] \tag{5.67}$$

and

$$\Lambda_{(\mathcal{B},\circledcirc)}[q_i \uplus q_j] = \Lambda_{(\mathcal{B},\circledcirc)}[q_i] \bigcup \Lambda_{(\mathcal{B},\circledcirc)}[q_i \triangleleft q_j] \tag{5.68}$$

which yield

$$\Lambda_{\circledcirc}[q_i \uplus q_j] = (\, \Lambda_{(\mathcal{S},\circledcirc)}[q_i \uplus q_j], \Lambda_{(\mathcal{B},\circledcirc)}[q_i \uplus q_j] \,) \tag{5.69}$$

the projection of $\Lambda_{\circledcirc}[q_j]$ into $\Lambda_{\circledcirc}[q_i]$, where the difference between the sets $\Lambda_{\circledcirc}[q_i \triangleleft q_j]$ and $\Lambda_{\circledcirc}[q_i \uplus q_j]$ is visualized in Fig. 5.11.

Projecting tuples from one extended sensorimotor neighborhood to another allows us establish motor connections between local sensorimotor patches which is a foundation for establishing a graph structure in $\mathcal{Q}^n$, where nodes correspond to neighborhoods and edges to said motor connections.

At the first glance it seems that the established reference frame transformation, as well as the projections require the configurations q_i and q_j to be known in order to be able to explicitly calculate $q_{ij} = q_i^{-1} \bullet q_i$. As however, controls $c \in C_{\odot}$ directly correspond to transformations $q \in \mathcal{Q}^n$ as given by $\tau_{\blacktriangleright}$ (see Eq. (4.17)), we can calculate the transformation between arbitrary configurations in $\mathcal{Q}^n$ as long as there exists a known control $c \in C_{\odot}$ that connects them. Moreover this connection is not limited to a single control, thus we can relate arbitrary configurations and thus arbitrary extended sensorimotor neighborhoods to each other as long as there exists

a sequence of known consecutive controls between them. Such a sequence is given by the previously established sensorimotor chain Ω_t (see Eq. (5.11) in Section 5.2.1) which consists of a list of t memory nodes, where each pair of consecutive memory nodes is connected by an effective control $c_i^{\otimes} \in C_{\odot}$.

As an example consider two consecutive memory nodes ω_i and ω_{i+1}. Naturally the control $c_i^{\otimes}$ corresponds to the transform

$$\tau_{\blacktriangleright}(c_i^{\otimes}) = q_{(i,i+1)} \tag{5.70}$$

which describes the transform from the configuration associated with ω_i to the configuration associated with ω_j as

$$q_{(\omega,i)} \bullet q_{(i,i+1)} = q_{(\omega,i)} \bullet \tau_{\blacktriangleright}(c_i^{\otimes}) = q_{(\omega,i+1)} \tag{5.71}$$

where correspondingly it holds that

$$\tau_{\blacktriangleright}(-c_i^{\otimes}) = q_{(i+1,i)} \tag{5.72}$$

with

$$q_{(\omega,i+1)} \bullet q_{(i+1,i)} = q_{(\omega,i+1)} \bullet \tau_{\blacktriangleright}(-c_i^{\otimes}) = q_{(\omega,i)}. \tag{5.73}$$

Applying $\tau_{\blacktriangleleft}$ yields

$$\tau_{\blacktriangleleft}(q_{(\omega,i)} \bullet q_{(i,i+1)}) = c_{(\omega,i+1)} \tag{5.74}$$

which is the control as an element of C_* that correspond to the direct transform between ω_i and ω_{i+1}. We now consider the general case, where ω_i, ω_j denote two arbitrary memory nodes from Ω_t. Given $\omega_i, \omega_j \in \Omega_t$ the sequence of effective controls that connects them is given by $C_{(\Omega,\odot)}[\omega_i, \omega_j]$ (see Eq. (5.14)). As each effective control is an element of $C_{\odot}$ we can create the corresponding transform by applying $\tau_{\blacktriangleright}$ which, applied to all elements from $C_{(\Omega,\odot)}[\omega_i, \omega_j]$ yields

$$\tau_{\blacktriangleright}(C_{(\Omega,\odot)}[\omega_i, \omega_j]) = \begin{cases} \tau_{\blacktriangleright}(c_i^{\otimes}), \tau_{\blacktriangleright}(c_{i+1}^{\otimes}), \cdots, \tau_{\blacktriangleright}(c_{j-1}^{\otimes}) & \text{if} \quad i < j \\ \tau_{\blacktriangleright}(-c_{i-1}^{\otimes}), \tau_{\blacktriangleright}(-c_{i-2}^{\otimes}), \cdots, \tau_{\blacktriangleright}(-c_j^{\otimes}) & \text{if} \quad i > j. \\ id & \text{if} \quad i = j. \end{cases} \tag{5.75}$$

Per definition it holds that $\forall c^{\otimes} : \tau_{\blacktriangleright}(c^{\otimes}) \in Q^n$, thus we can apply the group operation to these sequences. As Q^n is closed under the group operation (see Section 3.5) we know that successive application of $\bullet$ is guaranteed to be valid and will yield another element from Q^n. This element is q_{ij}, i.e., the group element describing the transformation between $q_{(\omega,i)}$ and $q_{(\omega,j)}$, thus

$$C_{(\Omega,\blacktriangleright)}[\omega_i, \omega_j] = \begin{cases} \tau_{\blacktriangleright}(c_i^{\otimes}) \bullet \tau_{\blacktriangleright}(c_{i+1}^{\otimes}) \bullet \cdots \bullet \tau_{\blacktriangleright}(c_{j-1}^{\otimes}) & \text{if} \quad i < j \\ \tau_{\blacktriangleright}(-c_{i-1}^{\otimes}) \bullet \tau_{\blacktriangleright}(-c_{i-2}^{\otimes}) \bullet \cdots \bullet \tau_{\blacktriangleright}(-c_j^{\otimes}) & \text{if} \quad i > j \\ id & \text{if} \quad i = j. \end{cases} \tag{5.76}$$

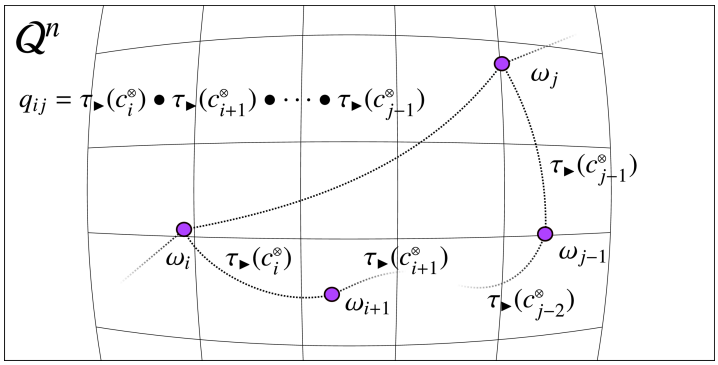


Figure 5.12: Calculation of the transformation between two arbitrary memory nodes ω_i and ω_j, where we assume that $i < j$. The sequence of controls that represent the path from ω_i to ω_j is given by the sensorimotor chain Ω_t as $C_{(\Omega,\circledcirc)}[\omega_i, \omega_j]$. To each effective control, the transformation can be calculated by applying $\tau_{\blacktriangleright}$. Thus, the transformation q_{ij} from $q_{(\omega,i)}$ to $q_{(\omega,j)}$ is given by $\tau_{\blacktriangleright}(c_i^{\otimes}) \bullet \tau_{\blacktriangleright}(c_{i+1}^{\otimes}) \bullet \cdots \bullet \tau_{\blacktriangleright}(c_{j-1}^{\otimes})$.

Therefore we can express the transform between arbitrary memory nodes as

$$q_{(\omega,i)} \bullet q_{ij} = q_{(\omega,i)} \bullet C_{(\Omega,\blacktriangleright)}[\omega_i, \omega_j] = q_{(\omega,j)}. \tag{5.77}$$

This transformation is relative to a global reference frame at $id \in Q^n$ as naturally $id \bullet q_{(\omega,i)} = q_{(\omega,i)}$ and $id \bullet q_{(\omega,j)} = q_{(\omega,j)}$. With respect to the reference frame at $q_{(\omega,i)}$, however, this yields

$$id \bullet q_{ij} = q_{ij} = C_{(\Omega,\blacktriangleright)}[\omega_i, \omega_j] = q_{(\omega,j)}. \tag{5.78}$$

Applying $\tau_{\blacktriangleleft}$ to Eq. (5.76) yields

$$\tau_{\blacktriangleleft}(C_{(\Omega,\blacktriangleright)}[\omega_i, \omega_j]) = \tau_{\blacktriangleleft}(q_{ij}) = c_{ij}, \tag{5.79}$$

where $c_{ij} \in C_*$. The sensorimotor chain thus provides a foundation to relate arbitrary memory nodes to each other and calculate the control c_{ij} as an element from C_* that corresponds to the direct transform from ω_i to ω_j and thus to the geodesic on Q^n. If it holds that $c_{ij} \in C_{\circledast}$ it is a valid control and can be executed by the agent's actuator. In cases where ω_i and ω_j are separated by one or more intermediate memory nodes, this control corresponds to a shortcut, i.e., a direct connection between the two memory nodes that does not require the agent to traverse through the intermediate nodes. The reference frame transformation for arbitrary memory nodes is visualized in Fig. 5.12. It has to be noted that despite c_i, c_j and c_{ij} being elements from $C \equiv \mathbb{R}^m$ the relation

$$\tau_{\blacktriangleleft}(\tau_{\blacktriangleright}(c_i) \bullet \tau_{\blacktriangleright}(c_j)) = c_i + c_j \tag{5.80}$$

only holds in cases where Q^n is an abelian group, i.e., a group for which $\bullet$ is commutative (see Section 3.5). A trivial example would be the Euclidean space $\mathbb{R}^n$, where the tangent space coincides with the group itself, group elements correspond to their tangent vectors and therefore adding vectors is valid there.

5.2.3.3 Sensorimotor Projection Sets

We now apply the previously reference transformations and projections given by Eqs. (5.66) and (5.69) to the extended sensorimotor sets $\Lambda_{\circledcirc}[q_i]$ and $\Lambda_{\circledcirc}[q_j]$. Let $\Lambda_{\circledcirc}[\omega_i]$ and $\Lambda_{\circledcirc}[\omega_j]$ be the two extended sensorimotor sets of the memory nodes $\omega_i, \omega_j \in Q^n$ respectively. The transformation from the origin of $\Lambda_{\circledcirc}[\omega_i]$ to the origin of $\Lambda_{\circledcirc}[\omega_j]$ is given as $C_{(\Omega,\blacktriangleright)}[\omega_i, \omega_j]$. The tuples $(c_{jk}, s_k) \in \Lambda_{(\mathcal{S},\circledcirc)}[\omega_j]$ and $(c_{jk}, \beta_k) \in \Lambda_{(\mathcal{B},\circledcirc)}[\omega_j]$ correspond to the configuration q_k in the vicinity of $q_{(\omega,j)}$, where $s_k \in \mathcal{S}$ has been perceived and $\beta_k \in \mathcal{B}$ is valid, respectively. The control $c_{jk} \in C_{\circledcirc}$ encodes the transformation from q_j to q_k as it holds that $q_{jk} = \tau_{\triangleright}(c_{jk})$. We combine both the transformation $C_{(\Omega,\blacktriangleright)}[\omega_i, \omega_j]$ from the origin of $\Lambda_{\circledcirc}[\omega_i]$ to the origin of $\Lambda_{\circledcirc}[\omega_j]$ and the transformation from the origin of $\Lambda_{\circledcirc}[\omega_j]$ to the configuration q_k and apply them to elements from $\Lambda_{\circledcirc}[\omega_j]$ (see Eq. (5.42)). We calculate the corresponding element from $C_{\circledcirc}$ by applying Eq. (4.38) which yields the transformed tuple from $\Lambda_{(\mathcal{S},\odot)}[\omega_j]$ as

$$\begin{aligned}(\tau_*(0_\odot), s_j) &\Rightarrow (\tau_{\blacktriangleleft}(C_{(\Omega,\blacktriangleright)}[\omega_i, \omega_j] \bullet \tau_{\triangleright}(\tau_*(0_\odot))), s_j) \\ &\Rightarrow (\tau_{\blacktriangleleft}(C_{(\Omega,\blacktriangleright)}[\omega_i, \omega_j] \bullet \tau_{\blacktriangleright}(0_\odot)), s_j) \\ &\Rightarrow (\tau_{\blacktriangleleft}(C_{(\Omega,\blacktriangleright)}[\omega_i, \omega_j] \bullet id), s_j) \\ &\Rightarrow (\tau_{\blacktriangleleft}(C_{(\Omega,\blacktriangleright)}[\omega_i, \omega_j]), s_j).\end{aligned} \tag{5.81}$$

To ensure that controls are within $C_{\circledcirc}$ we define

$$\lambda_s = \begin{cases} \{ (\tau_{\blacktriangleleft}(C_{(\Omega,\blacktriangleright)}[\omega_i, \omega_j]), s_j) \}, & \text{if} \quad \tau_{\blacktriangleleft}(C_{(\Omega,\blacktriangleright)}[\omega_i, \omega_j]) \in C_{\circledcirc} \\ \{ \}, & \text{otherwise.} \end{cases} \tag{5.82}$$

With respect to $\Lambda_{(\mathcal{B},\odot)}[\omega_j]$ we calculate

$$\begin{aligned}(\tau_*(0_\odot), \beta_\circ) &\Rightarrow (\tau_{\blacktriangleleft}(C_{(\Omega,\blacktriangleright)}[\omega_i, \omega_j] \bullet \tau_{\triangleright}(\tau_*(0_\odot))), \beta_\circ) \\ &\Rightarrow (\tau_{\blacktriangleleft}(C_{(\Omega,\blacktriangleright)}[\omega_i, \omega_j]), \beta_\circ)\end{aligned} \tag{5.83}$$

and

$$\begin{aligned}(\tau_*(c_j^{\ominus}), \beta_\bullet) &\Rightarrow (\tau_{\blacktriangleleft}(C_{(\Omega,\blacktriangleright)}[\omega_i, \omega_j] \bullet \tau_{\triangleright}(\tau_*(c_j^{\ominus}))), \beta_\bullet) \\ &\Rightarrow (\tau_{\blacktriangleleft}(C_{(\Omega,\blacktriangleright)}[\omega_i, \omega_j] \bullet \tau_{\blacktriangleright}(c_j^{\ominus})), \beta_\bullet).\end{aligned} \tag{5.84}$$

Accordingly we ensure that controls are within $C_{\circledcirc}$. Thus, we define

$$\lambda_\circ = \begin{cases} \{ (\tau_{\blacktriangleleft}(C_{(\Omega,\blacktriangleright)}[\omega_i, \omega_j]), \beta_\circ) \}, & \text{if} \quad \tau_{\blacktriangleleft}(C_{(\Omega,\blacktriangleright)}[\omega_i, \omega_j]) \in C_{\circledcirc} \\ \{ \}, & \text{otherwise} \end{cases} \tag{5.85}$$

and

$$\lambda_\bullet = \begin{cases} \{ (\tau_{\blacktriangleleft}(C_{(\Omega,\blacktriangleright)}[\omega_i, \omega_j] \bullet \tau_{\blacktriangleright}(c_j^{\ominus})), \beta_\bullet) \}, & \text{if} \quad \tau_{\blacktriangleleft}(C_{(\Omega,\blacktriangleright)}[\omega_i, \omega_j] \bullet \tau_{\blacktriangleright}(c_j^{\ominus})) \in C_{\circledcirc} \\ \{ \}, & \text{otherwise.} \end{cases} \tag{5.86}$$

From Eqs. (5.82), (5.85) and (5.86) we construct the individual reference frame transformations

$$\Lambda_{(\mathcal{S},\circledcirc)}[\omega_i \triangleleft \omega_j] = \lambda_s \tag{5.87}$$

and

$$\Lambda_{(\mathcal{B},\circledcirc)}[\omega_i \lhd \omega_j] = \begin{cases} \lambda_\circ \cup \lambda_\bullet, & \text{if} \quad c_j^{\ominus} \neq c_j^{\otimes} \\ \lambda_\circ, & \text{if} \quad c_j^{\ominus} = c_j^{\otimes} \quad \text{or} \quad j = t. \end{cases} \tag{5.88}$$

If $\omega_i = \omega_j$ it naturally holds that

$$\Lambda_{(\mathcal{S},\circledcirc)}[\omega_i \lhd \omega_j] = \Lambda_{(\mathcal{S},\circledcirc)}[\omega_i] = \Lambda_{(\mathcal{S},\circledcirc)}[\omega_j] \tag{5.89}$$

and correspondingly

$$\Lambda_{(\mathcal{B},\circledcirc)}[\omega_i \lhd \omega_j] = \Lambda_{(\mathcal{B},\circledcirc)}[\omega_i] = \Lambda_{(\mathcal{B},\circledcirc)}[\omega_j]. \tag{5.90}$$

We denote the tuple from Eqs. (5.87) and (5.88) the reference transformation

$$\Lambda_{\circledcirc}[\omega_i \lhd \omega_j] = (\, \Lambda_{(\mathcal{S},\circledcirc)}[\omega_i \lhd \omega_j], \Lambda_{(\mathcal{B},\circledcirc)}[\omega_i \lhd \omega_j] \,). \tag{5.91}$$

To account for the both the transformed elements from $\Lambda_{\circledcirc}[\omega_j]$ and the elements contained in the set $\Lambda_{\circledcirc}[\omega_i]$ we define

$$\Lambda_{(\mathcal{S},\circledcirc)}[\omega_i \uplus \omega_j] = \Lambda_{(\mathcal{S},\circledcirc)}[\omega_i] \bigcup \Lambda_{(\mathcal{S},\circledcirc)}[\omega_i \lhd \omega_j] \tag{5.92}$$

and

$$\Lambda_{(\mathcal{B},\circledcirc)}[\omega_i \uplus \omega_j] = \Lambda_{(\mathcal{B},\circledcirc)}[\omega_i] \bigcup \Lambda_{(\mathcal{B},\circledcirc)}[\omega_i \lhd \omega_j] \tag{5.93}$$

which yields

$$\Lambda_{\circledcirc}[\omega_i \uplus \omega_j] = (\, \Lambda_{(\mathcal{S},\circledcirc)}[\omega_i \uplus \omega_j], \Lambda_{(\mathcal{B},\circledcirc)}[\omega_i \uplus \omega_j] \,) \tag{5.94}$$

the projection of $\Lambda_{\circledcirc}[\omega_j]$ into $\Lambda_{\circledcirc}[\omega_i]$. The projection of the entire sensorimotor chain, i.e., the projection of the sensorimotor sample sets associated to all memory nodes of Ω_t to a given node ω_i is given by

$$\Lambda_{(\mathcal{S},\circledcirc)}[\omega_i \uplus \Omega_t] = \Lambda_{(\mathcal{S},\circledcirc)}[\omega_i] \bigcup_{1 \leq j \leq t,\, j \neq i} \Lambda_{(\mathcal{S},\circledcirc)}[\omega_i \lhd \omega_j] \tag{5.95}$$

and accordingly

$$\Lambda_{(\mathcal{B},\circledcirc)}[\omega_i \uplus \Omega_t] = \Lambda_{(\mathcal{B},\circledcirc)}[\omega_i] \bigcup_{1 \leq j \leq t,\, j \neq i} \Lambda_{(\mathcal{B},\circledcirc)}[\omega_i \lhd \omega_j], \tag{5.96}$$

where the combination of both yields

$$\Lambda_{\circledcirc}[\omega_i \uplus \Omega_t] = (\, \Lambda_{(\mathcal{S},\circledcirc)}[\omega_i \uplus \Omega_t], \Lambda_{(\mathcal{B},\circledcirc)}[\omega_i \uplus \Omega_t] \,), \tag{5.97}$$

the projection set of the sensorimotor chain Ω_t into the neighborhood of the memory node ω_i. Projecting Ω_t into a given neighborhoods $\Lambda_{\circledcirc}[\omega_i]$ enables us to represent all perceived information with respect to the agents-specific motor control reference frame centered in ω_i. This

transformation provides 'shortcuts' on Q^n as tuples from this set express perceptions at configurations in terms of the controls needed to directly reach them. In the following we label individual elements from Eqs. (5.95) and (5.96) as 'data points' given as

$$\lambda_{(\mathcal{S},j)} \in \Lambda_{(\mathcal{S},\circledcirc)}[\omega_i \uplus \Omega_t] \tag{5.98}$$

and

$$\lambda_{(\mathcal{B},j)} \in \Lambda_{(\mathcal{B},\circledcirc)}[\omega_i \uplus \Omega_t] \tag{5.99}$$

respectively.

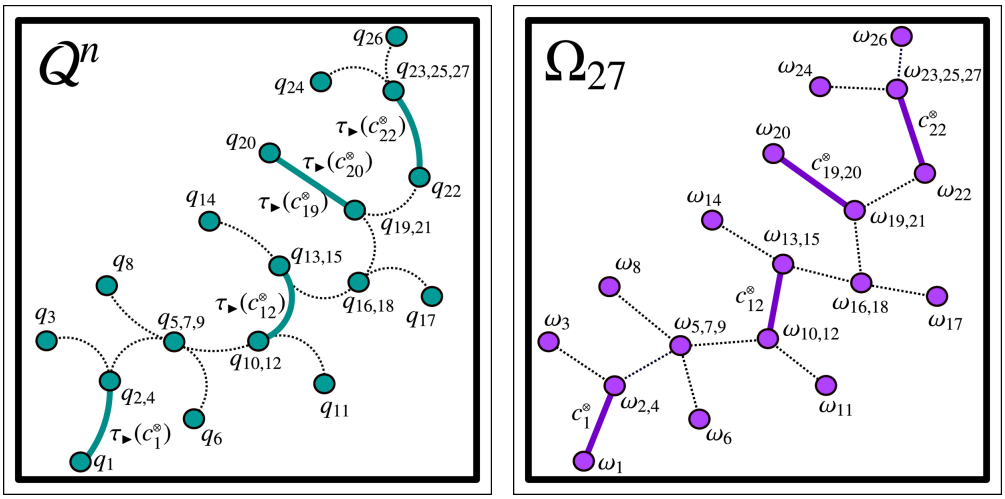
5.3 Building Sensorimotor Maps

In Section 5.2 we established the concept of both the sensorimotor chain and the sensorimotor neighborhoods. By combining both and defining transformations between arbitrary memory nodes we established a foundation for developing a graph-based structure on the configuration space. In this section we utilize these building blocks and establish the sensorimotor map and relate it to the intrinsic agent loop. Though ultimately our aim is to utilize the map for bootstrapping, i.e., for identifying the configuration space and the geometric interpretation of motor controls, we initially assume that both properties are known. Thus, both the matrix Lie group Q^n as well as the parameters for $\tau_{\blacktriangleright}$ and $\tau_{\blacktriangleleft}$ are given. In addition we consider the interaction with the environment to be noise free.

5.3.1 Motivational Example

As a motivational example we revisit the mobile robotic system depicted in Section 1.3 confronted with the task to navigate through its environment in order to find the configuration with the highest measurable temperature. For the sake of clarity only a subset of all configurations, transformations, memory nodes and effective controls will be explicitly labeled in the following figures. In addition the notation $q_{2,4,6}$ and $\omega_{2,4,6}$ will be used, indicating that the configurations q_2, q_4 and q_6 are identical, i.e., that the memory nodes ω_2, ω_4 and ω_6 correspond to the same identical configuration.

Task I The agent follows the same algorithm as described in Task I in Section 1.3: It measures the temperature, applies a random control, performs a second temperature measurement at the new location, and compares these two measurements. If the second temperature measurement is higher than the first, the agent has moved towards the heat source and repeats the aforementioned steps. If the second measured temperature is lower, the agent moved away from the heat source. It moves back to where it started by reversing the last control, re-enters the previous best configuration and starts with the first step. The algorithm terminates if after five iteration no configuration with a higher temperature has been found. After following this strategy for $t = 27$ steps, the agent reaches its final configuration, where the path it has traveled on Q^n, i.e., the individual configurations it traversed through and the transformations are depicted in Fig. 5.13a. An alternative



(a) Visualization of the configurations in Q^n associated to the memory nodes in Ω_{27}. Transformations between configurations correspond to effective controls on which $\tau_{\blacktriangleright}$ is applied.

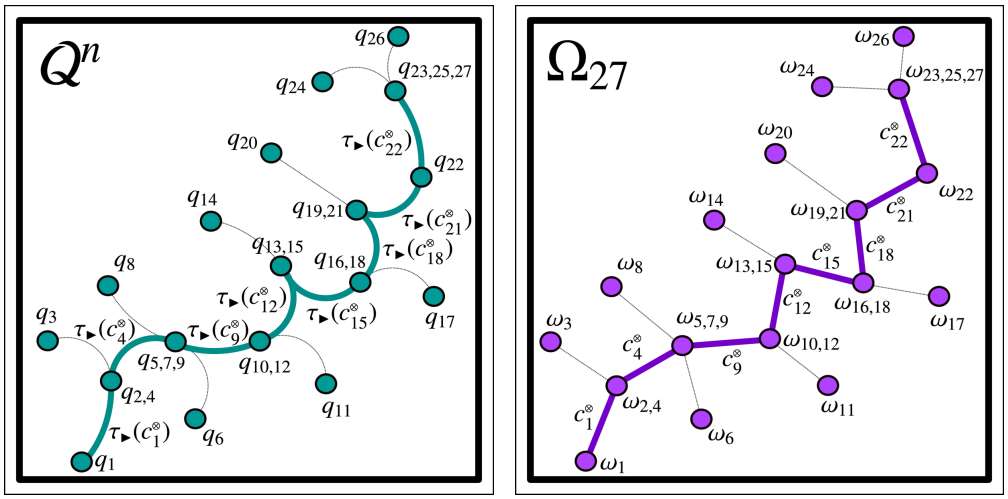
(b) Visualization of the memory nodes in Ω_{27}, where consecutive memory nodes are connected by the corresponding effective controls.

Figure 5.13: Visualization of the sensorimotor chain Ω_{27} after exploring the environment and reaching the configuration with the highest temperature. Figure 5.13a displays the configurations and transformations in Q^n, where Fig. 5.13b provides an abstract view on the memory nodes and their connections. For the sake of clarity, we only display selected transformations in Q^n and selected controls between memory nodes in Ω_{27}, respectively.

view is depicted in Fig. 5.13b that displays the sensorimotor chain Ω_{27} created from the alternating sequence of temperature perceptions and motor controls.

Task II We now switch the agent off, put it at its starting configuration, switch it on again and assign to it the task of moving to the heat source again. As previously established, a valid approach would be to simply repeat the entire sequence of previously executed controls. With respect to the sensorimotor chain Ω_{27} this would correspond to sequentially execute the controls $C_{(\Omega,\odot)}[\omega_1, \omega_{27}]$ (see Eq. (5.14)). However, as already established in Section 1.3, a better strategy would be to discard redundant controls. As subsequent redundant controls negate each other, a valid approach would be to check if it holds that for two consecutive controls from $C_{(\Omega,\odot)}[\omega_1, \omega_{27}]$ it holds that $c_i = -c_j$. If that is the case we can simply discard both and repeat this operation until no more redundant controls are found. The resulting path from q_1 to $q_{23,25,27}$, i.e., the path from ω_1 to $\omega_{23,25,27}$ is depicted in Fig. 5.14. Notably both navigational approaches did not rely on invoking $\tau_{\blacktriangleright}$ or $\tau_{\blacktriangleleft}$, i.e., neither re-tracing the entire sequence of previously executed controls nor following an accordingly optimized instance required any geometric interpretation.

Task III We now look at the implication of the final task, where the agent has the objective to find the shortest path from ω_1 to $\omega_{23,25,27}$. As a preparation we incorporate topological information into the sensorimotor chain, i.e., establish a connection between controls and paths on the configuration space by providing information on the Lie group Q^n and the



(a) Visualization of the transformations as elements from Q^n, leading from q_1 to $q_{23,25,27}$.

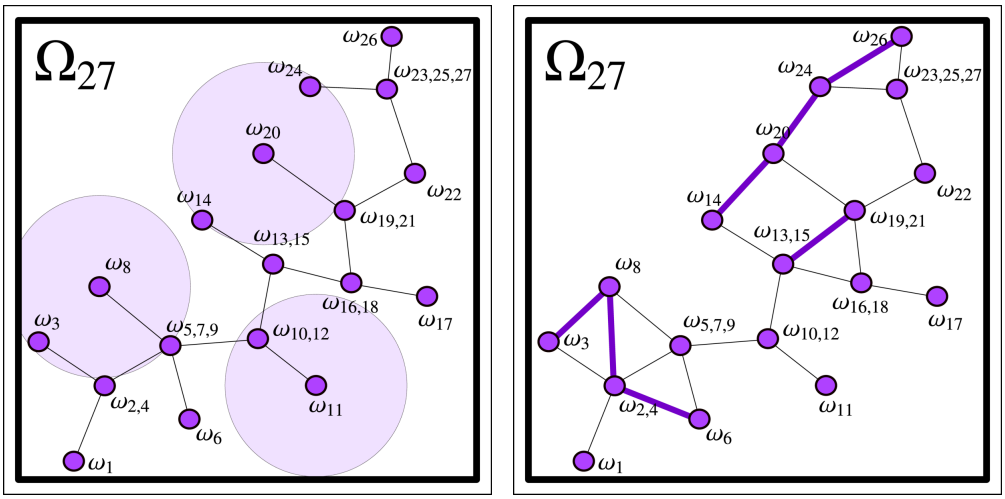
(b) Visualization of the sequence of effective controls as elements from $C_\odot$, describing the path from ω_1 to $\omega_{23,25,27}$.

Figure 5.14: Direct navigation on the sensorimotor chain Ω_{27}. Successive elimination of redundant controls from $C_{(\Omega,\odot)}[\omega_1, \omega_{27}]$ yields the sequence of effective controls $c^\otimes_1, c^\otimes_4, c^\otimes_9, \ldots, c^\otimes_{22}$ that describes the path from ω_1 to $\omega_{23,25,27}$, depicted in Fig. 5.14b. This corresponds to the transformations leading from q_1 to $q_{23,25,27}$ which are given as $\tau_\blacktriangleright(c^\otimes_1), \tau_\blacktriangleright(c^\otimes_4), \tau_\blacktriangleright(c^\otimes_9), \ldots, \tau_\blacktriangleright(c^\otimes_{22})$, depicted in Fig. 5.14a.

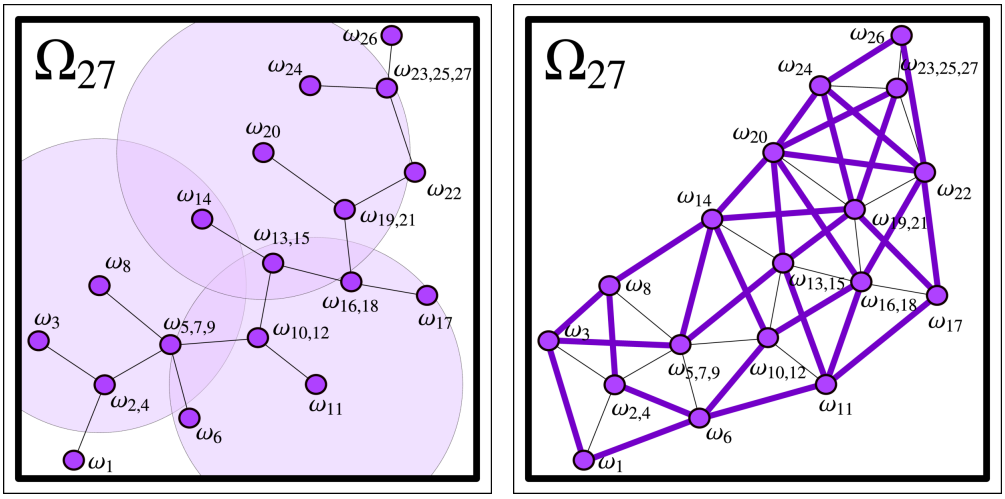
mapping from controls to the tangent space $T_{id}Q^n$. This allows us to utilize Eq. (5.97) and project Ω_{27} into the vicinity of each $\omega_i \in \Omega_{27}$. This projection establishes a reference frame at every ω_i, given by the extended sensorimotor neighborhood at the corresponding configuration $q_{(\omega,i)}$. This creates an overlapping set of local neighborhoods, where the location of a memory node ω_j within the reference frame of another node ω_i directly corresponds to the control that is required to move from ω_i to ω_j. Therefore these projected nodes directly encode shortcuts the agent can utilize to skip memory nodes. Thus, applying Eq. (5.97) to the sensorimotor chain creates a graph-structure, where memory nodes represent local patches on Q^n which are connected by edges that correspond to the controls required to traverse between them. As the size of the neighborhoods, i.e., the size of the local patches on Q^n depends on the radius of $C_\circledcirc$, the selection of $\epsilon_\circledcirc$ directly affects the topology of the graph which is visualized in Fig. 5.15.

The agent can utilize this graph for navigation by traversing from memory node to memory node, following the original or newly created edges until reaching its destination. Figure 5.16 visualizes two different paths from ω_1 to $\omega_{23,25,27}$ based on the two graphs depicted in Fig. 5.15.

It has to be noted, however, that in both of these cases we expect that the agent can follow all edges that have been created. Thus, for the sake of the argument, we assumed that $\forall c \in C_\circledcirc : c \in C_\circledast$, i.e., each control $c \in C_\circledcirc$ can be executed. This corresponds to the assumptions that the system is holonomic with $C = C_*$, and that all controls from C are valid with $C_\odot = C$.



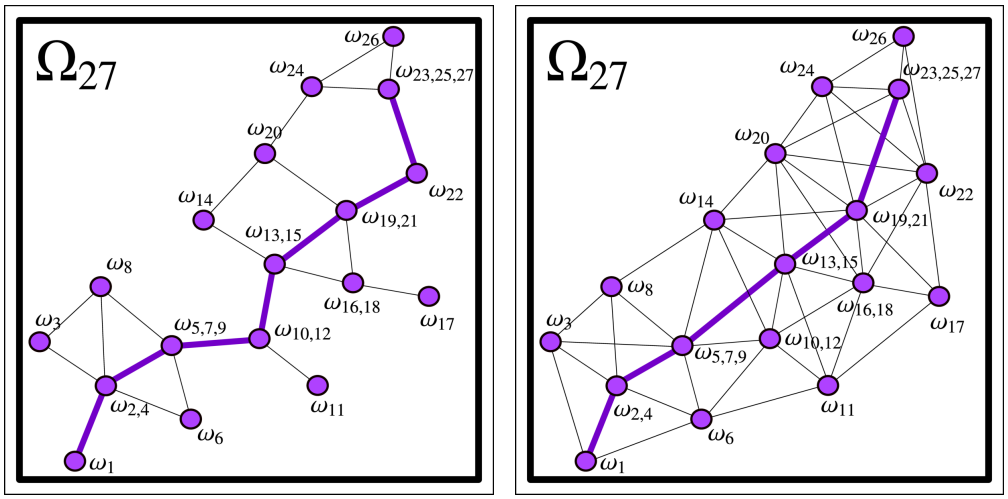
(a) Visualization of a small $\epsilon_{\circledcirc}$ with respect to Q^n and the induced graph created by calculating $\Lambda_{\circledcirc}[\omega_i \uplus \Omega_{27}]$ for each $\omega_i \in \Omega_{27}$. Neighborhood radius visualized for three selected memory nodes ω_8, ω_{11} and ω_{20}.



(b) Visualization of a large $\epsilon_{\circledcirc}$ with respect to Q^n and the induced graph created by calculating $\Lambda_{\circledcirc}[\omega_i \uplus \Omega_{27}]$ for each $\omega_i \in \Omega_{27}$. Neighborhood radius visualized for three selected memory nodes ω_8, ω_{11} and ω_{20}.

Figure 5.15: Visualization of the effect the radius $\epsilon_{\circledcirc}$ has on the structure of the graph created by calculating the projection sets $\Lambda_{\circledcirc}[\omega_i \uplus \Omega_{27}]$ for all memory nodes in the sensorimotor chain Ω_{27}. For the sake of clarity, neighborhoods are only visualized for an example set of memory nodes. Highlighted edges correspond to newly calculated controls from $C_{\circledcirc}$ that encode 'shortcuts' between memory nodes, i.e., between their associated configurations in Q^n.

Though constructing this graph structure is possible in the nonholonomic case as well, utilizing it for navigation is slightly more difficult. For a nonholonomic system the set of valid controls $C_{\circledast}$ is only embedded in C_*, thus shortcuts between memory nodes as elements from $C_{\circledcirc}$ are not guaranteed to be in $C_{\circledast}$. However, even if the calculated shortcuts can not be executed, they nevertheless provide information on the local patch on Q^n that is associated to the neighborhood



(a) Highlighted edges show the path from ω_1 to $\omega_{22,24,26}$. It is a sequence of controls from $C_\odot$ and $C_\circledcirc$, based on the graph created in Fig. 5.15a.

(b) Highlighted edges show the path from ω_1 to $\omega_{22,24,26}$. It is a sequence of controls from $C_\odot$ and $C_\circledcirc$, based on the graph created in Fig. 5.15b.

Figure 5.16: Visualization of paths on two different sensorimotor graphs, created using different values for the neighborhood radius $\epsilon_\circledcirc$. The paths that lead from ω_1 to $\omega_{22,24,26}$, utilize shortcuts on the graphs created by calculating the projection sets $\Lambda_\circledcirc[\omega_i \uplus \Omega_{27}]$ for all memory nodes $\omega_i \in \Omega_{27}$ using different neighborhood sizes.

of the memory node they were projected to. Thus, for any given memory node ω_i the projection $\Lambda_\circledcirc[\omega_i \uplus \Omega_t]$ represents a set of data points that describe a local patch on $\mathcal{Q}^n$, centered at the configuration $q_{(\omega,i)}$. This patch corresponds to the neighborhood of ω_i which establishes a control-specific and thus agent-specific coordinate frame. We thus represent the configuration space as a set of overlapping and connected neighborhoods, where the resulting graph covers the explored portion of the agent's environment.

5.3.2 The Sensorimotor Graph

The sensorimotor graph is a structure that decomposes the configuration space into local neighborhoods, where the origins of these neighborhoods correspond to the memory nodes. Attaching an extended sensorimotor neighborhood to each memory node ω_i from Ω_t allows us to project all information available within the entire sensorimotor chain into the individual sensorimotor sets. Thus, a given node represents the knowledge the agent has of a local patch of the configuration space. Utilizing Eq. (5.97), we denote the sensorimotor graph the set of all sensorimotor projection sets over all memory nodes of Ω_t as

$$\Lambda[\Omega_t] = \{\, \Lambda_\circledcirc[\omega_i \uplus \Omega_t] \,|\, \omega_i \in \Omega_t \,\}. \tag{5.100}$$

In the following we depict the algorithm to subsequently construct the sensorimotor graph, which is separated into two distinct functions Graph::Init() and Graph::Update() which are

explicit instantiations of the corresponding functions used in the intrinsic agent loop in Section 4.4. Thus, GRAPH::INIT() initializes the sensorimotor graph based on the initial sample $s_1 \in \mathcal{S}$, the matrix Lie group $\mathcal{Q}^n$ as well as the matrices $T_{\circlearrowleft}$ and $T_{\circlearrowright}$ (see Eq. (4.61)). Accordingly GRAPH::UPDATE() updates the graph with the tuple $(c^{\ominus}_{i-1}, c^{\otimes}_{i-1}, s_i)$, consisting of the scheduled and effective controls as well as the latest perceived sample (see Eq. (4.62)).

5.3.2.1 Sensorimotor Graph Initialization

Algorithm 3 depicts the function GRAPH::INIT(). The algorithm works as follows:

- Line 1: The algorithm requires $\epsilon_{\circledcirc}$ to be initialized as this parameter establishes the size of the neighborhoods, i.e., the radius of $C_{\circledcirc}$. Calling GRAPH::INIT() requires the initial sample $s_1 \in \mathcal{S}$ to be available to be able to initialize the first memory node. The configuration space is defined by $\mathcal{Q}^n$. The relation between the control space C and $\mathcal{Q}^n$ is defined by the matrices $T_{\circlearrowleft}$ and $T_{\circlearrowright}$, which parameterize the functions $\tau_{\blacktriangleright}$ and $\tau_{\blacktriangleleft}$, respectively (see Eqs. (4.17) and (4.38)). We provide $\mathcal{Q}^n$, $T_{\circlearrowleft}$ and $T_{\circlearrowright}$ to GRAPH::INIT() explicitly, to be able to utilize the sensorimotor graph with different parameterizations.

- Lines 2 and 3: The sensorimotor chain is initialized as an empty sequence and the sensorimotor graph is initialized as an empty set.

- Lines 4 and 5: Afterwards the initial memory node ω_1 is created and appended to the sensorimotor chain, where naturally ω_1 has no predecessor and no successor and only contains the initial sample s_1.

- Lines 6 to 8: The sample and obstruction sets associated to the memory node ω_1 are initialized, where $(\tau_*(0_{\odot}), s_1)$ links the initial sample to the initial configuration and $(\tau_*(0_{\odot}), \beta_{\circ})$ denotes that the initial configuration is unobstructed.

- Lines 9 to 11: The initial projection sets are created from the sample and obstruction sets associated to ω_1. These sets contain the initial data points $(0_{\circledcirc}, s_1)$ and $(0_{\circledcirc}, \beta_{\circ})$, given by the sample and obstruction neighborhoods associated with the initial memory node ω_1.

- Lines 12 and 13: The sensorimotor graph is updated by adding the projection set of the initial memory node. Afterwards the initialized sensorimotor graph is returned.

5.3.2.2 Sensorimotor Graph Update

Algorithm 4 depicts the function GRAPH::UPDATE(). The algorithm works as follows:

- Line 1: The algorithm requires the sensorimotor graph $\Lambda[\Omega_1]$ to be initialized, i.e., the function GRAPH::INIT() must haven been executed beforehand. Executing GRAPH::UPDATE() requires information on the last scheduled and effective controls $c^{\ominus}_{t-1}, c^{\otimes}_{t-1}$, as well as the latest perceived sample s_t, to be able to incorporate this information into the graph.

Sensorimotor Graph - Init

Require: $\epsilon_{\circledcirc}$ initialized from $\mathbb{R}^+$

1: **procedure** GRAPH::INIT($s_1, Q^n, T_{\cup}, T_{\cup}$)
▷ Initialize empty sensorimotor chain and empty sensorimotor map
2: $\Omega_0 \leftarrow ()$
3: $\Lambda[\Omega_0] \leftarrow \{\}$
▷ Create and initialize memory node ω_1 and append it to the sensorimotor chain
4: $\omega_1 \leftarrow (\{\}, \{s_1\}, \{\})$ ▷ Eq. (5.5)
5: $\Omega_1 \leftarrow \Omega_0 \oplus \omega_1$
▷ Initialize sensorimotor sample and obstruction sets for ω_1
6: $\Lambda_{(\mathcal{S},\circledcirc)}[\omega_1] \leftarrow \{(\tau_*(0_\odot), s_1)\}$ ▷ Eq. (5.44)
7: $\Lambda_{(\mathcal{B},\circledcirc)}[\omega_1] \leftarrow \{(\tau_*(0_\odot), \beta_\circ)\}$ ▷ Eq. (5.45)
8: $\Lambda_{\circledcirc}[\omega_1] \leftarrow (\Lambda_{(\mathcal{S},\circledcirc)}[\omega_1], \Lambda_{(\mathcal{B},\circledcirc)}[\omega_1])$ ▷ Eq. (5.46)
▷ Initialize sensorimotor projection sets for ω_1
9: $\Lambda_{(\mathcal{S},\circledcirc)}[\omega_1 \uplus \Omega_1] \leftarrow \Lambda_{(\mathcal{S},\circledcirc)}[\omega_1]$ ▷ Eq. (5.95)
10: $\Lambda_{(\mathcal{B},\circledcirc)}[\omega_1 \uplus \Omega_1] \leftarrow \Lambda_{(\mathcal{B},\circledcirc)}[\omega_1]$ ▷ Eq. (5.96)
11: $\Lambda_{\circledcirc}[\omega_1 \uplus \Omega_1] \leftarrow (\Lambda_{(\mathcal{S},\circledcirc)}[\omega_1 \uplus \Omega_1], \Lambda_{(\mathcal{B},\circledcirc)}[\omega_1 \uplus \Omega_1])$
▷ Assign sensorimotor projection set of ω_1 to sensorimotor graph
12: $\Lambda[\Omega_1] \leftarrow \Lambda[\Omega_0] \cup \{\Lambda_{\circledcirc}[\omega_1 \uplus \Omega_1]\}$ ▷ Eq. (5.100)
▷ Return current sensorimotor graph
13: **return** $\Lambda[\Omega_1]$
14: **end procedure**

Algorithm 3: Initializing the sensorimotor graph.

- Line 2: A new memory node is created and connected to its predecessor. The effective control $c^{\otimes}_{t-1}$ describes the transform from ω_{t-1} to ω_t. The connection from ω_t to ω_{t-1} is thus given by the inverted effective control $-c^{\otimes}_{t-1}$.

- Lines 3, 4 and 6: The previous memory node ω_{t-1} is connected to the latest memory node ω_t. This connection corresponds to the effective control $c^{\otimes}_{t-1}$ that describes the transform from ω_{t-1} to ω_t. If the agent encountered an obstruction, the scheduled and effective controls $c^{\ominus}_{t-1}$ and $c^{\otimes}_{t-1}$ differ. In this case, not only the effective, but the scheduled $c^{\ominus}_{t-1}$ control is stored to allow the agent to represent only partially executable controls, i.e., represent that executing $c^{\ominus}_{t-1}$ at the configuration $q_{(\omega,t-1)}$, associated to the memory node ω_{t-1}, is impossible.

- Line 8: The newly created memory node is appended to the sensorimotor chain.

- Lines 9 to 11: The sample set and the obstruction set associated to the memory node ω_t are updated, where $(\tau_*(0_\odot), s_t)$ links the latest sample to the current configuration and $(\tau_*(0_\odot), \beta_\circ)$ denotes that the current configuration is unobstructed.

- Lines 12 and 13: If the agent encountered an obstacle, the tuple $(\tau_*(c^{\ominus}_{t-1}), \beta_\bullet)$ is added to the obstruction set of the previous memory node, indicating that executing $c^{\ominus}_{t-1}$ at $q_{(\omega,t-1)}$ is impossible.

- Lines 15 to 17: The sensorimotor projection sets associated to ω_t are created as the union of its own sample and obstruction sets and the projection of all previous memory nodes. In this step, the entire sensorimotor chain Ω_t is folded into the neighborhood of ω_t. The sample set $\Lambda_{(\mathcal{S},\circledcirc)}[\omega_t \uplus \Omega_t]$ afterwards contains the sample data points associated directly to the memory node ω_t, and all sample data points associated to all other memory nodes that are within the neighborhood radius $\epsilon_\circledcirc$. The same holds for $\Lambda_{(\mathcal{B},\circledcirc)}[\omega_t \uplus \Omega_t]$, which contains all obstruction data points within the neighborhood of ω_t.

- Lines 18 to 21: After the new memory node has been updated, all other memory nodes are updated as well. However, we have only to account for the new information, given by the new sample s_t and the last scheduled and effective controls $c^{\ominus}_{t-1}, c^{\otimes}_{t-1}$. Sample information is encoded in the new memory node ω_t. To incorporate this information, the sensorimotor chain Ω_t is folded into the neighborhoods of all predecessors of ω_t, given as $\omega_1, \cdots, \omega_{t-1}$. However, only the sample associated to ω_t is used to update the respective sample projection sets. Obstruction sets have only to be updated if the agent encountered an obstruction in the last step. This information is stored in the memory node ω_{t-1} as the tuple of scheduled and effective controls $c^{\ominus}_{t-1}, c^{\otimes}_{t-1}$. To incorporate this information, the sensorimotor chain Ω_t is folded into the neighborhoods of all predecessors of ω_t, given as $\omega_1, \cdots, \omega_{t-1}$ The obstruction data associated to ω_{t-1} is used to update the obstruction projection sets of all memory nodes $\omega_1, \cdots, \omega_{t-1}$, which includes the projection set of ω_{t-1}. After these steps, all projection sets, i.e., all neighborhoods associated to all memory nodes in Ω_t are updated and contain all samples and obstructions within the radius $\epsilon_\circledcirc$.

- Lines 24 to 26: The sensorimotor graph is updated with the projection set that is associated to ω_t. Afterwards the sensorimotor graph is returned.

5.3.3 The Sensorimotor Map

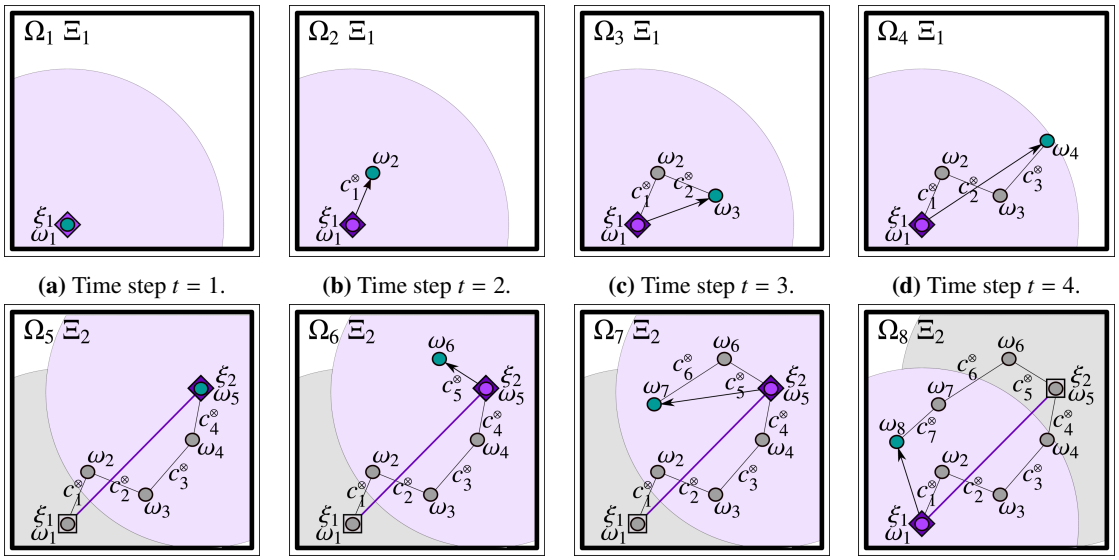
A disadvantage of the sensorimotor graph established in Algorithms 3 and 4 is that memory nodes are directly utilized as origins of reference frames. In every iteration of the agent control loop (see Algorithm 2), the sensorimotor chain is extended by one additional node and thus the number of reference frames is accordingly increased by one. As each neighborhood has to be updated with the new information perceived at each time step, the aim should be to keep the number of reference frames as low as possible.

The idea is to maintain the representation of the configuration space based on local neighborhoods, but reduce their number by detaching the reference frames associated with the memory nodes and instead add an additional abstraction layer between the graph-based representation of the configuration space and the sensorimotor chain. To this end, we create an approximately evenly-spaced tessellation of the configuration space, denoted as an anchor graph Ξ, which separates Q^n into overlapping neighborhoods. The origins of these neighborhoods are interconnected

Sensorimotor Graph - Update

Require: $\Lambda[\Omega_1]$ initialized
procedure GRAPH::UPDATE($c^{\ominus}_{t-1}, c^{\otimes}_{t-1}, s_t$)
▷ Create and initialize memory node ω_t associated to timestep t
$\omega_t \leftarrow (\{-c^{\otimes}_{t-1}\}, \{s_t\}, \{\})$ ▷ Eq. (5.4)
▷ Update connection between ω_{t-1} and ω_t
if $c^{\ominus}_{t-1} \neq c^{\otimes}_{t-1}$ **then**
$\overrightarrow{C}[\omega_{t-1}] \leftarrow \{c^{\ominus}_{t-1}, c^{\otimes}_{t-1}\}$ ▷ Eq. (5.3)
else
$\overrightarrow{C}[\omega_{t-1}] \leftarrow \{c^{\otimes}_{t-1}\}$ ▷ Eq. (5.3)
end if
▷ Append memory node ω_t to the sensorimotor chain
$\Omega_t \leftarrow \Omega_{t-1} \oplus \omega_t$
▷ Initialize sensorimotor sample and obstruction sets for ω_t
$\Lambda_{(\mathcal{S},\circledcirc)}[\omega_t] \leftarrow \{(\tau_*(0_\odot), s_t)\}$ ▷ Eq. (5.44)
$\Lambda_{(\mathcal{B},\circledcirc)}[\omega_t] \leftarrow \{(\tau_*(0_\odot), \beta_\circ)\}$ ▷ Eq. (5.45)
$\Lambda_{\circledcirc}[\omega_t] \leftarrow (\Lambda_{(\mathcal{S},\circledcirc)}[\omega_t], \Lambda_{(\mathcal{B},\circledcirc)}[\omega_t])$ ▷ Eq. (5.46)
▷ Update sensorimotor obstruction set for ω_{t-1}
if $c^{\ominus}_{t-1} \neq c^{\otimes}_{t-1}$ **then**
$\Lambda_{(\mathcal{B},\circledcirc)}[\omega_{t-1}] \leftarrow \Lambda_{(\mathcal{B},\circledcirc)}[\omega_{t-1}] \cup \{(\tau_*(c^{\ominus}_{t-1}), \beta_\bullet)\}$ ▷ Eq. (5.45)
end if
▷ Initialize sensorimotor projection sets for ω_t
$\Lambda_{(\mathcal{S},\circledcirc)}[\omega_t \uplus \Omega_t] \leftarrow \Lambda_{(\mathcal{S},\circledcirc)}[\omega_t] \cup_{1 \leq i < t} \Lambda_{(\mathcal{S},\circledcirc)}[\omega_t \lhd \omega_i]$ ▷ Eqs. (5.87) and (5.95)
$\Lambda_{(\mathcal{B},\circledcirc)}[\omega_t \uplus \Omega_t] \leftarrow \Lambda_{(\mathcal{B},\circledcirc)}[\omega_t] \cup_{1 \leq i < t} \Lambda_{(\mathcal{B},\circledcirc)}[\omega_t \lhd \omega_i]$ ▷ Eqs. (5.88) and (5.96)
$\Lambda_{\circledcirc}[\omega_t \uplus \Omega_t] \leftarrow (\Lambda_{(\mathcal{S},\circledcirc)}[\omega_t \uplus \Omega_t], \Lambda_{(\mathcal{B},\circledcirc)}[\omega_t \uplus \Omega_t])$ ▷ Eq. (5.97)
▷ Update sensorimotor sample projection sets for all ω_i with $i \neq t$ with new node ω_t
for all $\omega_i \in \Omega_t$ **where** $\omega_i \neq \omega_t$ **do**
$\Lambda_{(\mathcal{S},\circledcirc)}[\omega_i \uplus \Omega_t] \leftarrow \Lambda_{(\mathcal{S},\circledcirc)}[\omega_i \uplus \Omega_{t-1}] \cup \Lambda_{(\mathcal{S},\circledcirc)}[\omega_i \lhd \omega_t]$
▷ Update sensorimotor obstruction projection sets for all ω_i with $i \neq t$ with node ω_{t-1}
if $c^{\ominus}_{t-1} \neq c^{\otimes}_{t-1}$ **then**
$\Lambda_{(\mathcal{B},\circledcirc)}[\omega_i \uplus \Omega_t] \leftarrow \Lambda_{(\mathcal{B},\circledcirc)}[\omega_i \uplus \Omega_t] \cup \Lambda_{(\mathcal{B},\circledcirc)}[\omega_i \lhd \omega_{t-1}]$ ▷ Eqs. (5.88), (5.90) and (5.96)
end if
end for
▷ Assign sensorimotor projection set of ω_t to sensorimotor graph
$\Lambda[\Omega_t] \leftarrow \Lambda[\Omega_{t-1}] \cup \{\Lambda_{\circledcirc}[\omega_t \uplus \Omega_t]\}$ ▷ Eq. (5.100)
$t \leftarrow t + 1$
▷ Return current sensorimotor graph
return $\Lambda[\Omega_t]$
end procedure

Algorithm 4: Updating the sensorimotor graph.



(a) Time step $t = 1$. **(b)** Time step $t = 2$. **(c)** Time step $t = 3$. **(d)** Time step $t = 4$.

(e) Time step $t = 5$. **(f)** Time step $t = 6$. **(g)** Time step $t = 7$. **(h)** Time step $t = 8$.

Figure 5.17: Successive creation of the anchor graph Ξ over eight time steps. In each time step t the corresponding memory node ω_i is associated to the reference frame of the currently active anchor node $\xi \in \Xi$. If the agent leaves the neighborhood of an anchor node, the active reference frame is changed to another anchor node. Figure 5.17e visualizes the case where a new anchor node has to be created. Figure 5.17h visualizes the case in which an existing anchor node is re-used.

nodes that span a graph on the configuration space. Each of these anchor nodes $\xi_i \in \Xi$ allows to (a) establishes a local reference frame on $\mathcal{Q}^n$, given by an extended sensorimotor neighborhood, (b) maintains information on the relative transformation to any memory node in its vicinity, and (c) represent the current relative pose of the agent. An example is given in Fig. 5.17, that depicts the successive construction of the anchor graph. We assume that the agent executes the intrinsic agent loop, given in Algorithm 2, and that the agent successively executes a predefined set of scheduled controls. For each cycle, we examine the state of the anchor graph after invoking Init (Algorithm 2, Line 4) and Update (Algorithm 2, Line 10), respectively. We further assume, that in each cycle the sensorimotor chain Ω_t has already been updated, thus scheduled and effective of controls are available, as well as sample and obstruction data.

- Figure 5.17a: In time step $t = 1$. the anchor graph is created with an initial anchor node $\xi_1 \in \Xi_1$. It utilizes the sensorimotor neighborhood, provided by the initial memory node $\omega_1 \in \Omega_1$, to establish a local reference frame on $\mathcal{Q}^n$. The agent's initial pose at $q_1 \in \mathcal{Q}^n$ corresponds to the origin of this reference frame.

- Figure 5.17b: The agent executes the control $c_1^{\otimes}$, moving from q_1 to q_2. Information on q_2 is represented in the memory node ω_2 which is added to the reference frame at ξ_1. Notably, no sensorimotor neighborhood is created for ω_2 as it lies withing the neighborhood associated to ξ_1.

- Figures 5.17c and 5.17d: The agent executes two more controls, $c_2^{\otimes}$ and $c_3^{\otimes}$. As the agent is still within the neighborhood associated to ξ_1, only the agent's pose, i.e., the transformation to the latest memory node has to be updated in each step.
- Figure 5.17e: Executing $c_4^{\otimes}$, the control that moves the agent from q_4 to q_5, causes the agent to leave the neighborhood of ξ_1. As soon as the agent's distance from ξ_1 exceeds the neighborhood radius $\epsilon_{\circledcirc}$, a new anchor node ξ_2 is created, connected to ξ_1 and added to the anchor graph Ξ_2. It utilizes the sensorimotor neighborhood, provided by the latest memory node ω_5, to establish a new reference frame on Q^n. At this point, the agent transitions from one reference frame, given by ξ_1, to another. Thus, all further transformations are represented with respect to ξ_2.
- Figures 5.17f and 5.17g: The agent executes the controls $c_5^{\otimes}$ and $c_6^{\otimes}$. Information on memory nodes ω_6 and ω_7, as well as the agent's pose is represented with respect to ξ_2.
- Figure 5.17h: Executing $c_7^{\otimes}$ causes the agent to leave the neighborhood of ξ_2. However, the agent is within the neighborhood of ξ_1. Thus, instead of creating a new anchor node, the agent transitions from the reference frame of ξ_2 to the existing frame of ξ_1.

At this point 8 memory nodes and 2 anchor nodes have been created. The sensorimotor graph, given by Algorithms 3 and 4, would have instantiated a total of eight sensorimotor neighborhoods, one for each memory node. The proposed modification only creates two sensorimotor neighborhoods, while preserving the information given by the sensorimotor chain.

It has to be noted that, while the neighborhoods associated to ξ_1 and ξ_2 overlap, ξ_1 is not contained in the neighborhood of ξ_2 and vice versa. The degree of overlap between neighborhoods is determined by the relation between the neighborhood radius $\epsilon_{\circledcirc}$ and the maximal distance c_{max} the agent can travel within a single time step, where c_{max} naturally corresponds to the largest element from $C_{\odot}$. To guarantee that neighborhoods overlap, their radius has to be chosen as $\epsilon_{\circledcirc} > c_{\max}$. In the experiments described in Chapter 7, we chose $\epsilon_{\circledcirc}$ to be $\epsilon_{\circledcirc} = 1.5 \times c_{\max}$, guaranteeing the aforementioned overlap of neighborhoods.

In the following we will establish the notation used in the description of the sensorimotor map given in Algorithms 5 to 8. We denote the anchor graph as a set of u anchor nodes

$$\Xi_u = \{\xi_1, \xi_2, \cdots, \xi_u\}. \tag{5.101}$$

The number of anchor nodes is lower or equal to the number of memory nodes, thus $u \leq t$. An anchor node 'wraps' around one distinct memory node and utilizes its sensorimotor projection set and its reference frame. Other memory nodes are only represented with respect to their coordinates, but have no individual reference frame associated to them any more, as they only are used for representing the data (samples and obstructions) associated to them. To each anchor node ξ_i we associate one distinct memory node, thus

$$\forall \xi_i \in \Xi_u : \exists! \, \omega_j \in \Omega_t \quad \text{denoted} \quad \omega_{(\xi,i)}. \tag{5.102}$$

The extended sensorimotor set associated to an anchor node corresponds to the extended sensorimotor set of its associated memory node, as defined by Eq. (5.46), thus

$$\Lambda_{\circledcirc}[\xi_i] \equiv \Lambda_{\circledcirc}[\omega_{(\xi,i)}]. \tag{5.103}$$

Naturally, the projection of a sensorimotor set, associated to the memory node ω_j, into the sensorimotor set associated to the anchor node ξ_i, is defined according to Eqs. (5.87), (5.93) and (5.94) as

$$\Lambda_{(\mathcal{S},\circledcirc)}[\xi_i \uplus \omega_j] \equiv \Lambda_{(\mathcal{S},\circledcirc)}[\omega_{(\xi,i)}] \bigcup \Lambda_{(\mathcal{S},\circledcirc)}[\omega_{(\xi,i)} \lhd \omega_j], \tag{5.104}$$

$$\Lambda_{(\mathcal{B},\circledcirc)}[\xi_i \uplus \omega_j] \equiv \Lambda_{(\mathcal{B},\circledcirc)}[\omega_{(\xi,i)}] \bigcup \Lambda_{(\mathcal{B},\circledcirc)}[\omega_{(\xi,i)} \lhd \omega_j], \quad \text{and} \tag{5.105}$$

$$\Lambda_{\circledcirc}[\xi_i \uplus \omega_j] \equiv (\, \Lambda_{(\mathcal{S},\circledcirc)}[\xi_i \uplus \omega_j], \Lambda_{(\mathcal{B},\circledcirc)}[\xi_i \uplus \omega_j] \,). \tag{5.106}$$

Accordingly, the projection of the sensorimotor chain is defined with respect to Eqs. (5.95) to (5.97) as

$$\Lambda_{(\mathcal{S},\circledcirc)}[\xi_i \uplus \Omega_t] \equiv \Lambda_{(\mathcal{S},\circledcirc)}[\omega_{(\xi,i)} \uplus \Omega_t], \tag{5.107}$$

$$\Lambda_{(\mathcal{B},\circledcirc)}[\xi_i \uplus \Omega_t] \equiv \Lambda_{(\mathcal{B},\circledcirc)}[\omega_{(\xi,i)} \uplus \Omega_t], \quad \text{and} \tag{5.108}$$

$$\Lambda_{\circledcirc}[\xi_i \uplus \Omega_t] \equiv \Lambda_{\circledcirc}[\omega_{(\xi,i)} \uplus \Omega_t]. \tag{5.109}$$

We denote the set of all sensorimotor projections for all anchor nodes the sensorimotor map, as

$$\Lambda[\Xi_u] = \{\, \Lambda_{\circledcirc}[\xi i \uplus \Omega_t] \,|\, \xi_i \in \Xi_u \,\}. \tag{5.110}$$

To measure the distance between memory and anchor nodes, we define the function d that allows us to calculate the distance between two elements from $\Lambda_{\circledcirc}[\omega_i]$, i.e., the distance between their coordinates as elements from C_* with

$$d : C_* \times C_* \to \mathbb{R}^+. \tag{5.111}$$

This calculation is straightforward, as the group elements $q_{(c,i)}$ and $q_{(c,j)}$ are given by $q_{(c,i)} = \tau_{\blacktriangleright}(c_i)$ and $q_{(c,j)} = \tau_{\blacktriangleright}(c_j)$, respectively, and thus the distance corresponds to

$$d(c_i, c_j) = \|\tau_{\blacktriangleleft}(\tau_{\blacktriangleright}(c_i)^{-1} \bullet \tau_{\blacktriangleright}(c_j))\|. \tag{5.112}$$

In the following, we depict the algorithm to subsequently construct the sensorimotor map. It is separated into two distinct functions Map::Init() and Map::Update(), which are explicit instantiations of the corresponding functions used in the intrinsic agent loop in Section 4.4. Thus, Map::Init() initializes the sensorimotor map based on the initial sample $s_1 \in \mathcal{S}$, the matrix Lie group Q^n, as well as the matrices $T_{\circlearrowleft}$ and $T_{\circlearrowright}$ (see Eq. (4.61)). Correspondingly, Map::Update() updates the map with the tuple $(c^{\ominus}_{i-1}, c^{\otimes}_{i-1}, s_i)$ of scheduled and effective controls as well as the latest perceived sample (see Eq. (4.62)). To utilize the sensorimotor map as an objective function in Chapter 6, we define the function Map::Rebuild() that takes an existing sensorimotor map and rebuilds its graph structure, based on a different parameterization, i.e., a different matrix Lie group and/or different projection matrices.

5.3.3.1 Sensorimotor Map Initialization

Algorithm 5 depicts the function MAP::INIT(). The algorithm works as follows:

- Line 1: The algorithm requires $\epsilon_{\circledcirc}$ to be initialized as this parameter established the size of the neighborhoods, i.e., the radius of $C_{\circledcirc}$. Calling MAP::INIT() requires the initial sample $s_1 \in \mathcal{S}$ to be available to be able to initialize the first memory node. The configuration space is defined by Q^n. The relation between the control space C and Q^n is defined by the matrices $T_{\circlearrowleft}$ and $T_{\circlearrowright}$, which parameterize the functions $\tau_{\blacktriangleright}$ and $\tau_{\blacktriangleleft}$ respectively (see Eqs. (4.17) and (4.40)). We provide Q^n, $T_{\circlearrowleft}$ and $T_{\circlearrowright}$ to MAP::INIT() explicitly, to be able to utilize the sensorimotor map in the bootstrapping scenario, given in Chapter 6, where we will instantiate it with varying parameters.

- Lines 2 to 4: In addition to initializing the sensorimotor chain as an empty sequence, the anchor graph and the sensorimotor map are both initialized as empty sets.

- Lines 5 to 9: Like in GRAPH::INIT (see Algorithm 3), the initial memory node ω_1 is created and appended to the sensorimotor chain Ω_1. Afterwards its sample and obstruction sets are initialized.

- Lines 10 to 12: The initial anchor node ξ_1 is created and the initial memory node ω_1 is associated to it. The anchor node ξ_1 'wraps' around the memory node ω_1 and uses its sensorimotor projection set and its reference frame. The anchor graph is then updated with the initial anchor node.

- Line 13: The agent's initial pose corresponds to the configuration associated to the initial memory node ω_1. As the anchor node ξ_1 utilizes the reference frame of ω_1, the agent's initial pose $q_1 \in Q^n$ corresponds to the origin of this reference frame. Thus, ξ_1 is set to be the current active reference frame.

- Lines 14 to 16: The initial projection sets are created from the sample and obstruction sets associated to ξ_1 as given by Eqs. (5.107) to (5.109).

- Line 17 The sensorimotor map is updated by adding the initial projection set to it. Afterwards the initialized sensorimotor map is returned.

5.3.3.2 Sensorimotor Map Update

Algorithms 6 and 7 depicts the function MAP::UPDATE(). The algorithm works as follows:

- Line 1: The algorithm requires the sensorimotor map $\Lambda[\Xi_1]$ to be initialized, i.e., the function MAP::INIT() must haven been executed beforehand. Executing MAP::UPDATE() requires information on the last scheduled and effective controls $c^{\ominus}_{t-1}, c^{\otimes}_{t-1}$, as well as the latest perceived sample s_t, to be able to incorporate this information into the map.

Sensorimotor Map - Init

Require: $\epsilon_{\circledcirc}$ initialized from $\mathbb{R}^+$

1: **procedure** MAP::INIT($s_1, \mathcal{Q}^n, T_{\cup}, T_{\cup}$)
▷ Initialize empty sensorimotor chain and empty anchor graph
2: $\Omega_0 \leftarrow ()$
3: $\Xi_0 \leftarrow \{\}$
▷ Initialize empty sensorimotor map
4: $\Lambda[\Xi_0] \leftarrow \{\}$
▷ Create and initialize memory node ω_1 and append it to sensorimotor chain
5: $\omega_1 \leftarrow (\{\}, \{s_1\}, \{\})$ ▷ Eq. (5.5)
6: $\Omega_1 \leftarrow \Omega_0 \oplus \omega_1$
▷ Initialize sensorimotor sample and obstruction sets for ω_t
7: $\Lambda_{(\mathcal{S},\circledcirc)}[\omega_1] \leftarrow \{(\tau_*(0_\odot), s_1)\}$ ▷ Eq. (5.44)
8: $\Lambda_{(\mathcal{B},\circledcirc)}[\omega_1] \leftarrow \{(\tau_*(0_\odot), \beta_\circ)\}$ ▷ Eq. (5.45)
9: $\Lambda_{\circledcirc}[\omega_1] \leftarrow (\Lambda_{(\mathcal{S},\circledcirc)}[\omega_1], \Lambda_{(\mathcal{B},\circledcirc)}[\omega_1])$ ▷ Eq. (5.46)
▷ Create and initialize anchor node ξ_1 and assign it to the anchor graph
10: $u \leftarrow 1$
11: $\xi_1 \leftarrow \omega_1$ ▷ Eq. (5.102)
12: $\Xi_1 \leftarrow \Xi_0 \cup \{\xi_1\}$
▷ Set current reference frame to ξ_1
13: $\xi_{ref} \leftarrow \xi_1$
▷ Initialize sensorimotor projection sets for ξ_1
14: $\Lambda_{(\mathcal{S},\circledcirc)}[\xi_1 \uplus \Omega_t] \leftarrow \Lambda_{(\mathcal{S},\circledcirc)}[\omega_{(\xi,1)}]$ ▷ Eq. (5.107)
15: $\Lambda_{(\mathcal{B},\circledcirc)}[\xi_1 \uplus \Omega_t] \leftarrow \Lambda_{(\mathcal{B},\circledcirc)}[\omega_{(\xi,1)}]$ ▷ Eq. (5.108)
16: $\Lambda_{\circledcirc}[\xi_1 \uplus \Omega_t] \leftarrow (\Lambda_{(\mathcal{S},\circledcirc)}[\xi_1 \uplus \Omega_t], \Lambda_{(\mathcal{B},\circledcirc)}[\xi_1 \uplus \Omega_t])$ ▷ Eq. (5.109)
▷ Assign sensorimotor projection set of ξ_1 to sensorimotor map
17: $\Lambda[\Xi_1] \leftarrow \Lambda[\Xi_0] \cup \{\Lambda_{\circledcirc}[\xi_1 \uplus \Omega_t]\}$ ▷ Eq. (5.110)
▷ Return current sensorimotor map
18: **return** $\Lambda[\Xi_1]$
19: **end procedure**

Algorithm 5: Initializing the sensorimotor map.

- Lines 2 to 4, 6 and 8 to 13: Like in GRAPH::UPDATE (see Algorithm 4), a new memory node is created, connected to its predecessor and added to the sensorimotor chain. The sample and obstruction sets of this newly created memory node are then updated with the latest perceived sample s_t and the information on the unobstructed current configuration.

- Lines 15 and 16: The anchor node $\xi_{min} \in \Xi_u$ that is closest to the current memory node ω_t is identified. If it holds that ω_t is within the Euclidean n-ball associated to an existing anchor node, it is within the associated neighborhood and this anchor node is utilized as the new local reference frame. Otherwise, a new anchor node is created that represents a different local patch on the configuration space $\mathcal{Q}^n$.

- Lines 17 to 20: The current memory node ω_t is associated to a newly created anchor node ξ_u which is then set to be the active reference frame and added to the anchor graph.
- Lines 21 to 24: If a new anchor node ξ_u is created, its sensorimotor projection sets are initialized by folding the sensorimotor chain Ω_t into the neighborhood of the associated memory node $\omega_{(\xi,u)}$. The sample set $\Lambda_{(\mathcal{S},\circledcirc)}[\xi_u \uplus \Omega_t]$ afterwards contains the sample data points associated directly to the memory node $\omega_{(\xi,u)}$ and all sample data points associated to all other memory nodes that are within the neighborhood radius $\epsilon_\circledcirc$. The same holds for $\Lambda_{(\mathcal{B},\circledcirc)}[\xi_u \uplus \Omega_t]$ which contains all obstruction data points within the neighborhood of $\omega_{(\xi,u)}$. Afterwards, the sensorimotor map is updated by adding the newly created sensorimotor projection set to it.
- Line 26: The anchor node closest to the latest memory node ω_t is set to be the active reference frame.
- Lines 27 to 30: As the sensorimotor projection sets associated to ξ_{ref} already contain all data associated to the memory nodes $\omega_1, \cdots, \omega_{t-1}$, we only have to update it with the information associated to the new node ω_t.
- Lines 33 to 37: All anchor nodes, except ξ_{ref} that has already been accounted for, have to be updated with the information associated to the new node ω_t. To that end, the sensorimotor chain Ω_t is folded into the neighborhoods of all anchor nodes $\xi_i \in \Xi_u \setminus \xi_{ref}$. However, only the sample associated to ω_t is used to update the respective sample projection sets, while obstruction sets have only to be updated if the agent encountered an obstruction in the last step. This information is stored in the memory node ω_{t-1} as the tuple of scheduled and effective controls $c^{\ominus}_{t-1}, c^{\otimes}_{t-1}$. To incorporate this information, the sensorimotor chain Ω_t is folded into the neighborhoods of all anchor nodes. $\xi_i \in \Xi_u \setminus \xi_{ref}$. After these steps, all projection sets, i.e., all neighborhoods associated to all anchor nodes in Ξ_u are updated and contain all samples and obstructions within the radius $\epsilon_\circledcirc$.
- Lines 40 and 41: The updated sensorimotor map is returned.

5.3.3.3 Sensorimotor Map Rebuilding

In order to utilize the sensorimotor map in the bootstrapping scenario in Chapter 6, we need to be able to rebuild the map with a different matrix Lie group and/or modified projection matrices. To that end, we define the function MAP::REBUILD() that takes an existing sensorimotor map, extracts the sensorimotor chain as a hypothesis-invariant foundation and rebuilds the graph structure based on the provided matrix Lie group and the corresponding projection matrices. Algorithm 8 depicts the function MAP::REBUILD() which works as follows:

- Line 1: The algorithm requires $\Lambda[\Xi_u]$ to be initialized, i.e., MAP::INIT() must haven been executed beforehand. Executing MAP::REBUILD() requires the current map $\Lambda[\Xi_u]$ and the parameters required for rebuilding it. These parameters are the matrix Lie group Q^n that defines the new configuration space, and the corresponding projection matrices $T_\cup$ and $T_\cup$ that define the mapping between the control space C and the manifold.

Sensorimotor Map - Update (1/2)

Require: $\Lambda[\Xi_u]$ initialized
1: **procedure** MAP::UPDATE($c^{\ominus}_{t-1}, c^{\otimes}_{t-1}, s_t$)
▷ Create and initialize memory node ω_t associated to timestep t
2: $\omega_t \leftarrow (\{-c^{\otimes}_{t-1}\}, \{s_t\}, \{\})$ ▷ Eq. (5.4)
▷ Update connection between ω_{t-1} and ω_t
3: **if** $c^{\ominus}_{t-1} \neq c^{\otimes}_{t-1}$ **then**
4: $\overrightarrow{C}[\omega_{t-1}] \leftarrow \{c^{\ominus}_{t-1}, c^{\otimes}_{t-1}\}$ ▷ Eq. (5.3)
5: **else**
6: $\overrightarrow{C}[\omega_{t-1}] \leftarrow \{c^{\otimes}_{t-1}\}$
7: **end if**
▷ Append memory node ω_t to sensorimotor chain
8: $\Omega_t \leftarrow \Omega_{t-1} \oplus \omega_t$
▷ Initialize sensorimotor sample and obstruction sets for ω_t
9: $\Lambda_{(\mathcal{S},\circledcirc)}[\omega_t] \leftarrow \{(\tau_*(0_\odot), s_t)\}$ ▷ Eq. (5.44)
10: $\Lambda_{(\mathcal{B},\circledcirc)}[\omega_t] \leftarrow \{(\tau_*(0_\odot), \beta_\circ)\}$ ▷ Eq. (5.45)
11: $\Lambda_{\circledcirc}[\omega_t] \leftarrow (\Lambda_{(\mathcal{S},\circledcirc)}[\omega_t], \Lambda_{(\mathcal{B},\circledcirc)}[\omega_t])$ ▷ Eq. (5.46)
▷ Update sensorimotor obstruction set for ω_{t-1}
12: **if** $c^{\ominus}_{t-1} \neq c^{\otimes}_{t-1}$ **then**
13: $\Lambda_{(\mathcal{B},\circledcirc)}[\omega_{t-1}] \leftarrow \Lambda_{(\mathcal{B},\circledcirc)}[\omega_{t-1}] \cup \{(\tau_*(c^{\ominus}_{t-1}), \beta_\bullet)\}$ ▷ Eq. (5.45)
14: **end if**
(continued in Algorithm 7)

Algorithm 6: Updating the sensorimotor map (1/2).

- Line 2: The sensorimotor chain is extracted from the sensorimotor map $\Lambda[\Xi_u]$. As established in Section 5.2.1, the sensorimotor chain Ω_t does not contain any geometric interpretation and solely depicts the experienced sensorimotor interaction between the agent and its environment. Thus, it contains all data required to construct a new sensorimotor map from.

- Line 3: A new sensorimotor map is initialized by calling MAP::INIT() (Algorithm 5). This function is provided with the sample s_1 associated to the first memory node ω_1 and the new parameters Q^n, $T_{\circlearrowleft}$, $T_{\circlearrowleft}$.

- Lines 4 to 6: For each memory node in the sensorimotor chain Ω_t, the information associated to it is provided to the newly created sensorimotor map by calling MAP::UPDATE() (Algorithms 6 and 7). Thus, we successively build a new sensorimotor map from 'recorded data', i.e., from the existing sensorimotor chain Ω_t.

- Line 8: After the data associated to all memory nodes $\omega_i \in \Omega_t$ has been processed, the sensorimotor map is returned.

Sensorimotor Map - Update (2/2)

(continuation from Algorithm 6)

▷ Retrieve anchor node closest to ω_t

$\xi_{min} \leftarrow \arg\min_{\xi_i \in \Xi_u} d(\omega_t, \omega_{(\xi,i)})$ with $i \neq j$

▷ Depending on distance to ω_t either create new anchor node or reuse existing one

if $d(\omega_t, \omega_{(\xi,min)}) > \epsilon_{\circledcirc}$ **then**

▷ Create and initialize new anchor node ξ_u and assign it to the anchor graph

$u \leftarrow u + 1$

$\xi_u \leftarrow \omega_t$ ▷ Eq. (5.102)

$\Xi_u \leftarrow \Xi_{u-1} \cup \xi_u$

▷ Set current reference frame to ξ_u

$\xi_{ref} \leftarrow \xi_u$

▷ Initialize sensorimotor projection sets for ξ_u

$\Lambda_{(\mathcal{S},\circledcirc)}[\xi_u \uplus \Omega_t] \leftarrow \Lambda_{(\mathcal{S},\circledcirc)}[\omega_{(\xi,u)}] \cup_{1 \leq i < t} \Lambda_{(\mathcal{S},\circledcirc)}[\omega_{(\xi,u)} \lhd \omega_i]$ ▷ Eqs. (5.87) and (5.107)

$\Lambda_{(\mathcal{B},\circledcirc)}[\xi_u \uplus \Omega_t] \leftarrow \Lambda_{(\mathcal{B},\circledcirc)}[\omega_{(\xi,u)}] \cup_{1 \leq i < t} \Lambda_{(\mathcal{B},\circledcirc)}[\omega_{(\xi,u)} \lhd \omega_i]$ ▷ Eqs. (5.88) and (5.108)

$\Lambda_{\circledcirc}[\xi_u \uplus \Omega_t] \leftarrow (\, \Lambda_{(\mathcal{S},\circledcirc)}[\xi_u \uplus \Omega_t], \Lambda_{(\mathcal{B},\circledcirc)}[\xi_u \uplus \Omega_t] \,)$

▷ Assign sensorimotor projection set of ξ_u to sensorimotor map

$\Lambda[\Xi_u] \leftarrow \Lambda[\Xi_{u-1}] \cup \{\, \Lambda_{\circledcirc}[\xi_u \uplus \Omega_t] \,\}$ ▷ Eq. (5.110)

else

▷ Reuse existing node, set current reference frame to ξ_{min}

$\xi_{ref} \leftarrow \xi_{min}$

▷ Update sensorimotor projection sets for ξ_{ref} with ω_t

$\Lambda_{(\mathcal{S},\circledcirc)}[\xi_{ref} \uplus \Omega_t] \leftarrow \Lambda_{(\mathcal{S},\circledcirc)}[\xi_{ref} \uplus \Omega_{t-1}] \cup \Lambda_{(\mathcal{S},\circledcirc)}[\omega_{(\xi,ref)} \lhd \omega_t]$ ▷ Eqs. (5.87) and (5.107)

$\Lambda_{(\mathcal{B},\circledcirc)}[\xi_{ref} \uplus \Omega_t] \leftarrow \Lambda_{(\mathcal{B},\circledcirc)}[\xi_{ref} \uplus \Omega_{t-1}] \cup \Lambda_{(\mathcal{B},\circledcirc)}[\omega_{(\xi,ref)} \lhd \omega_t]$ ▷ Eqs. (5.88) and (5.107)

▷ Update sensorimotor obstruction projection set of ξ_{ref} with ω_{t-1}

if $c^{\ominus}_{t-1} \neq c^{\otimes}_{t-1}$ **then**

$\Lambda_{(\mathcal{B},\circledcirc)}[\xi_{ref} \uplus \Omega_t] \leftarrow \Lambda_{(\mathcal{B},\circledcirc)}[\xi_{ref} \uplus \Omega_t] \cup \Lambda_{(\mathcal{B},\circledcirc)}[\omega_{(\xi,ref)} \lhd \omega_{t-1}]$

end if

end if

▷ Update sensorimotor projection sets for all $\xi_i \neq \xi_{ref}$ with new node ω_t

for all $\xi_i \in \Xi_u$ **where** $\xi_i \neq \xi_{ref}$ **do**

$\Lambda_{(\mathcal{S},\circledcirc)}[\xi_i \uplus \Omega_t] \leftarrow \Lambda_{(\mathcal{S},\circledcirc)}[\xi_i \uplus \Omega_{t-1}] \cup \Lambda_{(\mathcal{S},\circledcirc)}[\omega_{(\xi,i)} \lhd \omega_t]$ ▷ Eqs. (5.104) and (5.107)

$\Lambda_{(\mathcal{B},\circledcirc)}[\xi_i \uplus \Omega_t] \leftarrow \Lambda_{(\mathcal{B},\circledcirc)}[\xi_i \uplus \Omega_{t-1}] \cup \Lambda_{(\mathcal{B},\circledcirc)}[\omega_{(\xi,i)} \lhd \omega_t]$ ▷ Eqs. (5.105) and (5.108)

▷ Update sensorimotor obstruction projection sets for all $\xi_i \neq \xi_{ref}$ with node ω_{t-1}

if $c^{\ominus}_{t-1} \neq c^{\otimes}_{t-1}$ **then**

$\Lambda_{(\mathcal{B},\circledcirc)}[\xi_i \uplus \Omega_t] \leftarrow \Lambda_{(\mathcal{B},\circledcirc)}[\xi_i \uplus \Omega_t] \cup \Lambda_{(\mathcal{B},\circledcirc)}[\omega_{(\xi,i)} \lhd \omega_{t-1}]$

end if

end for

$t \leftarrow t + 1$

▷ Return current sensorimotor map

return $\Lambda[\Xi_u]$

end procedure

Algorithm 7: Updating the sensorimotor map (2/2).

Sensorimotor Map - Rebuild

Require: $\Lambda[\Xi_u]$ initialized
1: **procedure** MAP::REBUILD($\Lambda[\Xi_u]$, Q^n, $T_\cup$, $T_\cup$)
▷ Retrieve sensorimotor chain from provided map
2: $\Omega_t \leftarrow$ extracted from existing $\Lambda[\Xi_u]$
▷ Initialize new sensorimotor map with initial sample
3: $\Lambda[\Xi_1] \leftarrow$ MAP::INIT($s_{(\omega,1)}, Q^n$, $T_\cup$, $T_\cup$) ▷ Algorithm 5
4: **for** each memory node $\omega_i \in \Omega_t$ with $i > 1$ **do**
▷ Retrieve scheduled and effective controls from ω_{i-1} to ω_i
5: $c^{\ominus}_{i-1}, c^{\otimes}_{i-1} \leftarrow \overrightarrow{C}[\omega_{i-1}]$
▷ Update sensorimotor map
6: $\Lambda[\Xi_u] \leftarrow$ MAP::UPDATE($c^{\ominus}_{i-1}$, $c^{\otimes}_{i-1}$, $s_{(\omega,i)}$) ▷ Algorithms 6 and 7
7: **end for**
▷ Return rebuild sensorimotor map
8: return $\Lambda[\Xi_u]$
9: **end procedure**

Algorithm 8: Rebuilding a sensorimotor map with new manifold and projection matrices.

5.4 Summary

In this chapter we established the sensorimotor chain Ω_t as a domain-independent structure that encodes the interaction between the agent and its environment. We utilized the properties of the configuration space Q^n as a matrix Lie group to establish sensorimotor neighborhoods $\Lambda_\odot[q]$ with $q \in Q^n$, local patches on the configuration space that are represented with respect to the agent's motor capabilities. Combining these two concepts allowed us to associate a geometric interpretation to the sensorimotor chain and to eventually identify shortcuts in the configuration space. These shortcuts allowed us to develop the sensorimotor map Ξ_u as a graph-based representation of the configuration space, where local patches, $\Lambda[\xi_i]$ with $\xi_i \in \Xi_u$, are connected to each other, each endowed with their own reference frame.

In this way the sensorimotor map is closely related to the original definition of a manifold, established in Section 3.3. A manifold M can be defined as atlas, a set of tuples (U_α, ϕ_α) consisting of a neighborhood $U_\alpha \subset M$ and a corresponding coordinate function ϕ_α that maps this neighborhood into an open subset of $\mathbb{R}^n$. For two overlapping charts U_α, U_β a transition map can be constructed that allows to perform coordinate transformations as given by Eq. (3.15).

In our case the manifold corresponds to the configuration space Q^n, i.e., a matrix Lie group. Each extended sensorimotor neighborhood associated to an anchor node $\xi \in \Xi_u$ represents a chart on Q^n, where the coordinate function and its inverse are given by $\tau_\blacktriangleleft$ and $\tau_\triangleright$ respectively. The former maps elements from Q^n into $C_* \equiv \mathbb{R}^n$, the latter elements from $C_* \equiv \mathbb{R}^n$ into Q^n.

Extended sensorimotor neighborhoods are constructed in such a way that they overlap, i.e., that coordinate transformations between elements from one neighborhood to another are possible. As the sensorimotor map $\Lambda[\Xi_u]$ is composed of all these overlapping neighborhoods, we can interpret it as an atlas-like representation of the explored portion of the configuration space.

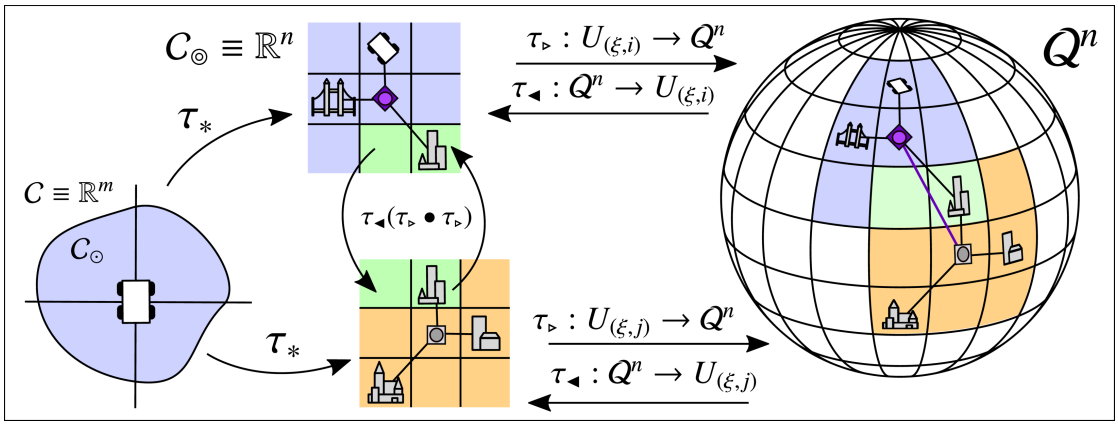


Figure 5.18: The sensorimotor map, interpreted as a parameterizable atlas-like structure. The functions $\tau_{\triangleleft}$ and $\tau_{\triangleright}$ provide a mapping between elements of Q^n and elements of $C_* \equiv \mathbb{R}^m$. Overlapping neighborhoods of distinct configurations on Q^n, associated to individual anchor nodes $\xi \in \Xi_u$, are mapped to local Euclidean reference frames, given by $C_{\circledcirc} \subseteq \mathbb{R}^n$.

Figure 5.18 visualizes this similarities, by relating the sensorimotor map and its building blocks to Fig. 3.4.

A difference between the manifold definition given in Section 3.3 and the sensorimotor map is that the coordinate functions introduced in Section 3.3 are fixed, where in the case of the sensorimotor map, they are parameterized by the matrices $T_{\circlearrowleft}$ and $T_{\circlearrowleft}$ (see Eqs. (4.36) and (4.37) and Eqs. (4.12) and (4.13) respectively). The sensorimotor map thus can be seen as a 'parameterizable' atlas, where Q^n defines the underlying structure, and the choice of $T_{\circlearrowleft}$ and $T_{\circlearrowleft}$ determines how individual charts are spatially related to each other.

In the next chapter, we utilize this parameterization for bootstrapping as it allows us to utilize the sensorimotor map as an objective function for determining a parameterization that causes the individual charts to align in a consistent way.

6

Bootstrapping using Sensorimotor Maps

In this chapter, we utilize the sensorimotor map developed in Chapter 5 for bootstrapping, i.e., for inferring the sensorimotor properties of the configuration space based on the history of past interactions between the agent and its environment that is encoded in the sensorimotor chain. The structure is as follows: In Section 6.2.1 we propose a parameterization of the sensorimotor map that allows us to systematically create maps based on real-valued vectors. In Section 6.2.2, we introduce the concept of intrinsic conflict, a measure that allows us to compute a positive real-valued number for each sensorimotor map that indicates how well the map and thus the corresponding parameterization, matches the information encoded in the sensorimotor chain. We establish the connection between the intrinsic conflict function and nonparametric regression algorithms and propose two kernel-based approaches that allow us to utilize the sensorimotor map in conjunction with the conflict function as an objective function. We then propose two views on the bootstrapping scenario. In Section 6.3.1, we propose an algorithm for bootstrapping, interpreting it as a classification task. In Section 6.3.2, we interpret it as an optimization problem and propose a corresponding algorithm based on particle swarm optimization. In Section 6.4, we introduce the concept of exploration patterns as predefined control sequences and extend the intrinsic agent loop introduced in Section 4.4, Algorithm 2 to utilize these patterns. In Section 6.5, we integrate both bootstrapping approaches into two distinct versions of the previously established intrinsic agent loop. The chapter concludes in Section 6.6 with a summary, a discussion, and an outlook on the next chapter.

6.1 Synopsis

In the last chapter we developed the sensorimotor map as a representation that creates a tesselation of the explored configuration space by establishing a set of connected neighborhoods where each neighborhood represents information on samples and obstructions with respect to a local

Euclidean reference frame, given by the agent's motor controls. This information is provided by the sensorimotor chain, which abstractly represents the past interaction between the agent and its environment. In this context, we assumed that the valid parameterization of the sensorimotor map was known, i.e., that the matrix Lie group and the matrices that defined the relation between the extended control space and the group's tangent space were given.

In this chapter, however, the parameterization is assumed to be unknown. Instead of establishing the sensorimotor map as a representation for the information that is encoded in the sensorimotor chain, we utilize the map as an evaluation tool in order to find a suitable parameterization that is consistent with the sequence of past sensorimotor experiences.

As a first step, we interpret different parameterizations of the sensorimotor map as a hypothesis, a tuple consisting of (a) the matrix Lie group that defines the assumed structure of the configuration space and (b) the matrices that encode the assumed relation between the group's tangent space and the agent's motor controls. We relate these matrices to the tangent space of the general linear group $GL(n, \mathbb{R})$ (see Section 3.5) and the tangent space of the special orthogonal group $SO(n)$ (see Appendix A.1), respectively, which allows us to represent these matrices as real-valued vectors.

In a second step, we establish a function that allows us to assess to which extent a sensorimotor map induced by a distinct parameterization matches the information encoded in the sensorimotor chain. Such a parameterization induces a fixed geometric meaning to motor controls as it maps each control to an element of the matrix Lie group. Thus, it assigns a transformation matrix to each control that describes the effect this control has on the agent's position and orientation. Applying this geometric interpretation to the entire sensorimotor chain, a sequence of memory nodes connected by controls, 'folds' the chain in a hypothesis-specific way. This folding analogy is inspired by the domain of protein folding, introduced in Section 2.3.5. Thus, in the following we use the term 'folding' with respect to the aforementioned geometric interpretation As each memory node corresponds to a distinct configuration in Q^n, folding the sensorimotor chain can be seen as folding the configuration space itself, i.e., inducing hypothesis-specific spatial relations between individual configurations. These spatial relations between memory nodes are naturally applied to the information associated to them, namely to the elements from the sample set $\mathcal{S}$ and the boundary set $\mathcal{B}$. As established in Section 4.2, we assume a locally smooth relation between the configuration space and perceivable samples. Naturally, a valid parameterization of the sensorimotor chain induces a folding that causes the data points to be arranged in such a way that their locally smooth properties are preserved. An invalid parameterization, however, induces an invalid folding that causes incorrect relative spatial relations between the data points and introduces discontinuity, i.e., disorder in the otherwise locally smooth representation. This locally smooth structure existing on Q^n is preserved by mapping from Q^n into the extended sensorimotor neighborhoods in the sensorimotor map $\Lambda[\Xi_u]$. Thus, we can perform the calculations within these neighborhoods, which provides us with a local Euclidean reference frame. This allows us to utilize nonparametric regression algorithms to establish local models that describe the functional relation between the set of valid controls $C_\odot$, $\mathcal{S}$ and $\mathcal{B}$ as induced by Q^n. Based on these models, we establish a measure for intrinsic conflict, which relates the average generalization error to the amount of disorder in the underlying set of data points.

Combining the parameterization of the sensorimotor map with the conflict function allows us to systematically create hypotheses and evaluate them with respect to the intrinsic conflict. We initially treat the bootstrapping task as a classification problem, thus as a scenario where we have a to determine the true hypothesis from a given a set of hypotheses with varying parameters. Naturally, the hypothesis with the minimal conflict is the one that explains the data given by the sensorimotor chain best. We then consider a different approach, interpreting the bootstrapping task as an optimization problem utilizing a derivative-free optimization algorithm to minimize the conflict function and thus find a parameterization that induces a minimal amount of intrinsic conflict.

6.2 Preliminary Definitions

In this section we establish the foundations for the two bootstrapping extensions of the intrinsic agent loop that are proposed in Section 6.4. In a first step we introduce hypotheses as tuples of a matrix Lie group and a projection matrix and establish a functional relation between real-valued vectors and these matrices where the aim is to freely parameterize sensorimotor maps. In a second step we establish a measure of inconsistency, i.e., a measure of intrinsic conflict that allows us to evaluate sensorimotor maps with respect to the question on whether they are a valid representation of the data provided by the sensorimotor chain. We then combine both concepts and propose two algorithms that allow us to find a suitable parameterization with minimal intrinsic conflict, the former being a classification approach the latter an optimization approach.

6.2.1 Parameterization of the Sensorimotor Map

Our first objective is to find a way to systematically create sensorimotor maps, i.e., define a parameterization that allows us to establish a functional relation between a z-dimensional real-valued vector and the matrices mapping elements from C_* to $T_{id}\mathcal{Q}^n$ and vice versa (see Eqs. (4.36) and (4.37) and Eqs. (4.12) and (4.13) respectively). Though, in theory, $\mathcal{Q}^n$, as well as the projection matrices $T_{\cup}$ and $T_{\cup}$, can be arbitrarily chosen, the matrix dimensions naturally depend on the dimension of the manifold. Thus we initially assume an arbitrary but fixed hypothesis matrix Lie group to be given as $\mathcal{Q}^n_{\mathcal{H}}$ where n is selected to be equal to (holonomic case) or greater than (nonholonomic case) the degree of freedom of the agent's actuator. A hypothesis h_i is given as the tuple

$$h_i = (\mathcal{Q}^n_{(\mathcal{H},i)}, T_{(\cup,i)}, T_{(\cup,i)}). \tag{6.1}$$

that consists of a Lie group $\mathcal{Q}^n$ and accordingly selected matrices $T_{\cup}$ and $T_{\cup}$ where naturally it holds that $T_{\cup} = T_{\cup}^{-1}$ (see Section 4.3.3.2). We denote a set of k distinct hypotheses as

$$\mathcal{H}_k = \{\, h_1,\, h_2,\, \ldots,\, h_k \,\}. \tag{6.2}$$

Our aim is now to parameterize hypotheses by defining a function

$$\Upsilon[\mathcal{Q}^n_{\mathcal{H}}] : \mathbb{R}^z \to GL(n, \mathbb{R}) \tag{6.3}$$

that maps from a z-dimensional vector to an invertible $n \times n$ matrix. Naturally, as the dimension of $T_{\circlearrowleft}$ and $T_{\circlearrowright}$ depend on the dimension of $\mathcal{Q}^n$, we need to make this function dependent on the matrix Lie group if creates the projection matrices for. As $T_{\circlearrowleft}$ and $T_{\circlearrowright}$ are invertible matrices we can utilize the fact that the set of invertible $n \times n$ matrices (in conjunction with the group operation) is in itself a matrix Lie group, namely the general linear group $GL(n, \mathbb{R})$ (see Eq. (3.50)). Thus an approach for calculating such a parameterization would be to select a vector with n^2 elements, convert it into a $n \times n$ matrix and hope for the best, i.e., hope that this matrix is an element of $GL(n, \mathbb{R})$. Though chances are pretty small that columns of an arbitrarily selected matrix of random numbers are linearly dependent, we look for a solution where the invertibility is not a matter of chance but instead is guaranteed. Luckily we can exploit the fact that $GL(n, \mathbb{R})$ is a matrix Lie group and thus the exponential map and the matrix logarithm allow us to map elements from its tangent space $T_{id}GL(n, \mathbb{R})$ to the group and vice versa. The tangent space of $GL(n, \mathbb{R})$ has the same dimension as $GL(n, \mathbb{R})$ itself and therefore corresponds to the set of $n \times n$ matrices, however without the restriction of these matrices having to be invertible. Thus utilizing the tangent space of $GL(n, \mathbb{R})$ for Eq. (6.3) yields Eq. (6.4) which maps elements from $\mathbb{R}^{n^2}$ to $n \times n$-matrices, followed by applying the matrix exponential to them that maps elements from $T_{id}GL(n, \mathbb{R})$ to $GL(n, \mathbb{R})$. The function is given as

$$\Upsilon[\mathcal{Q}^n_{\mathcal{H}}]^{\oplus} : \mathbb{R}^z \rightarrow GL(n, \mathbb{R}) \quad \text{with} \quad z = n^2 \tag{6.4}$$

which translates to

$$\begin{aligned} \Upsilon[\mathcal{Q}^n_{\mathcal{H}}]^{\oplus}([r_1, r_2, \ldots, r_{n^2}]^T) &= exp \begin{bmatrix} r_1 & r_1 & \cdots & r_n \\ r_{n+1} & r_{n+2} & \cdots & r_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ r_{n(n-1)+1} & r_{n(n-1)+2} & \cdots & r_{n^2} \end{bmatrix} \\ &= T_{\circlearrowleft}, \end{aligned} \tag{6.5}$$

where $[r_1, r_2, \ldots, r_{n^2}]^T \in \mathbb{R}^{n^2}$ and, as required, $T_{\circlearrowleft} \in GL(n, \mathbb{R})$. A potential issue with this parameterization is that the number of free parameters z corresponds to the squared number of dimensions of the hypothesis matrix Lie group $\mathcal{Q}^n$ which in this case matches the number of generators of $GL(n, \mathbb{R})$. As an example consider $SE(3)$, the 6-dimensional matrix Lie group describing translation and rotation in $\mathbb{R}^3$ (see Appendix A.2.2). Parameterizing the mapping from ($C_* \equiv \mathbb{R}^6$) to $T_{id}SE(3)$ would require utilizing elements from $GL(6, \mathbb{R})$ which are 6×6 matrices having to be encoded as vectors from $\mathbb{R}^{36}$.

To limit the number of parameters, instead of utilizing $GL(n, \mathbb{R})$, we use $SO(n)$, which is the special orthogonal group of dimension n that describes the rotation of objects in n-dimensional Euclidean space (see Appendix A.1). As $SO(n)$ is a subgroup of $GL(n, \mathbb{R})$ it holds that $\forall g \in SO(n) : g \in GL(n, \mathbb{R})$. Elements from $SO(n)$ are invertible $n \times n$ matrices where the number of generators corresponds to $z = \binom{n}{2}$ with $n > 1$ (see Appendix A.1). Thus utilizing the tangent space of $SO(n)$ for Eq. (6.3) yields Eq. (6.6) which maps elements from $\mathbb{R}^z$ to elements from $T_{id}SO(n)$, followed by applying the matrix exponential to them that maps these to $SO(n)$. The function is given as

$$\Upsilon[\mathcal{Q}^n_{\mathcal{H}}]^{\ominus} : \mathbb{R}^z \rightarrow SO(n) \quad \text{with} \quad z = \binom{n}{2} \quad \text{given} \quad n > 1 \tag{6.6}$$

which translates to

$$\Upsilon[Q^n_{\mathcal{H}}]^{\ominus}([r_1, r_2, \ldots, r_z]^T) = \exp(r_1\hat{\mathfrak{g}}_1 + r_1\hat{\mathfrak{g}}_2 + \cdots + r_z\hat{\mathfrak{g}}_z) = T_{\circlearrowleft}. \tag{6.7}$$

Here $[r_1, r_2, \ldots, r_z]^T$ denotes a vector from $\mathbb{R}^z$, the parameters $\hat{\mathfrak{g}}_1, \cdots, \hat{\mathfrak{g}}_z$ denote the generators of $SO(n)$ and $T_{\circlearrowleft}$ is an invertible matrix as an element from $SO(n)$. The number of parameters required to encode a projection matrix for $SE(3)$ in this way would correspond to the number of generators for $SO(6)$ with is $z = 15$.

It has to be noted that the decision which parameterization to choose, depends on the requirements. While $\Upsilon[Q^n_{\mathcal{H}}]^{\oplus}$ has a larger parameter space, the number of matrices it can represent covers the entirety of $GL(n, \mathbb{R})$. Using $\Upsilon[Q^n_{\mathcal{H}}]^{\ominus}$ requires fewer parameters but, can only represent matrices that correspond to n-dimensional rotations. Thus, $\Upsilon[Q^n_{\mathcal{H}}]^{\oplus}$ should be used in cases where the mapping between $C_{\odot}$ and $T_{id}Q^n$ allows controls to be scaled and skewed. The parameterization $\Upsilon[Q^n_{\mathcal{H}}]^{\ominus}$ should be used in cases where the mapping from $C_{\odot}$ and $T_{id}Q^n$ only rotates the coordinate frame of $C_{\odot}$, but does not scale or skew the controls internally.

We now relate Eq. (6.4) and Eq. (6.6) to the sensorimotor map $\Lambda[\Xi_u]$. The sensorimotor map depends on the sensorimotor chain Ω_t, as the hypothesis-invariant representation of the agent's sensorimotor interaction with the environment, and a hypothesis h_i, which encodes the structure of the configuration space and the geometric effects of the agent's motor controls. Thus we define

$$\Xi : \Omega^*_t \times \mathcal{H} \to \Lambda^*[\Xi], \tag{6.8}$$

where Ω^*_t denotes the set of all possible sensorimotor chains with length t, $\mathcal{H}$ denotes the set of all possible hypotheses, and $\Lambda^*[\Xi]$ denotes the set of all possible sensorimotor maps. The functional interpretation of the sensorimotor map (see Algorithms 5 to 8), allows us to view it as a mapping from a set of parameters to a graph structure, and ultimately allows us to assign a real-valued number to it that indicates its intrinsic conflict. For the sake of clarity we have omitted $\epsilon_{\circledcirc}$, the radius of $C_{\circledcirc}$ as given in Eq. (4.10). Representing Eq. (6.8) solely with respect to the parameterization of the matrices $T_{\circlearrowleft}$ and $T_{\circlearrowleft}$ yields

$$(\Xi \circ \Upsilon)[\Omega_t, Q^n_{\mathcal{H}}] : \mathbb{R}^z \to \Lambda[\Xi_u] \tag{6.9}$$

where Ω_t and $Q^n_{\mathcal{H}}$ are given instances of a sensorimotor chain and matrix Lie group and $\mathbb{R}^z$ is the only free parameter. Thus, Eq. (6.9) describes the sensorimotor map as a Ω_t- and $Q^n_{\mathcal{H}}$-specific function of $\mathbb{R}^z$ where naturally, with respect to Eqs. (6.4) and (6.6), we define

$$(\Xi \circ \Upsilon^{\oplus})[\Omega_t, Q^n_{\mathcal{H}}] : \mathbb{R}^z \to \Lambda[\Xi_u] \quad \text{with} \quad z = n^2 \tag{6.10}$$

and correspondingly

$$(\Xi \circ \Upsilon^{\ominus})[\Omega_t, Q^n_{\mathcal{H}}] : \mathbb{R}^z \to \Lambda[\Xi_u] \quad \text{with} \quad z = \binom{2}{n} \quad \text{given} \quad n > 1. \tag{6.11}$$

6.2.2 Intrinsic Consistency and Conflict in Sensorimotor Neighborhoods

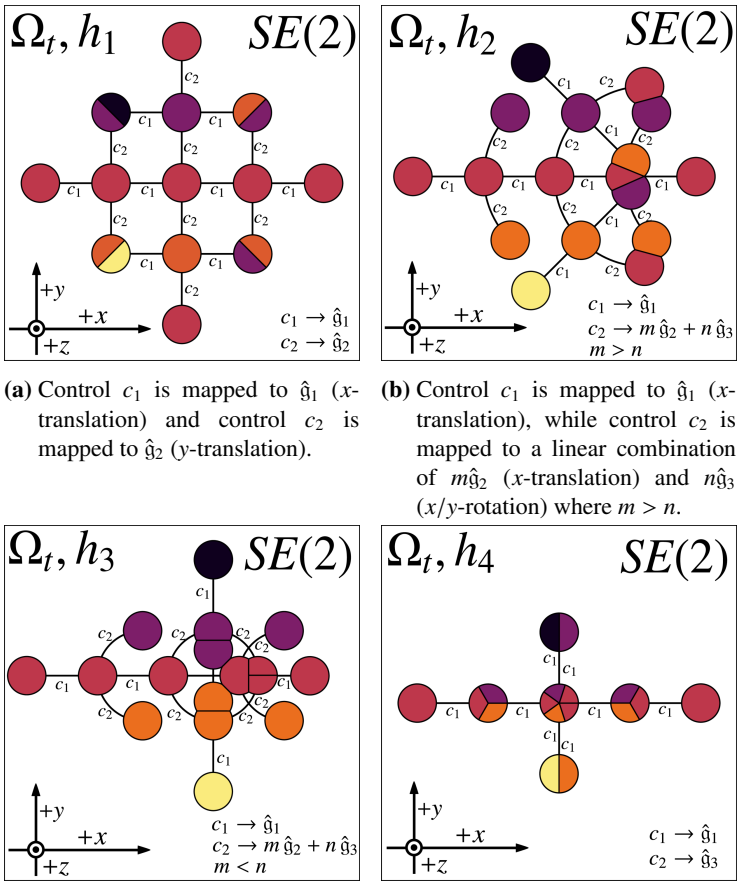
Our aim in this section is to develop a method for evaluating a given sensorimotor map created from a sensorimotor chain Ω_t and a hypothesis h_i according to Eqs. (6.10) and (6.11) and assess the validity of this hypothesis, i.e., assess to which extend h_i matches a valid matrix Lie group and a valid parameterization. Thus we want to establish a function

$$\Delta : \Lambda[\Xi_u] \to \mathbb{R}^+ \tag{6.12}$$

that assigns to a given sensorimotor map a real number, indicating whether the map is intrinsically 'sound'. The challenging aspect is that we want to evaluate and validate a given hypothesis based on a sensorimotor map, where the parametrization that defines the intrinsic geometry of said sensorimotor map is given by the hypothesis we want to evaluate. As the geometric relations between individual sample and obstruction data points $\lambda_{(\mathcal{S})}, \lambda_{(\mathcal{B})}$ (see Eqs. (5.98) and (5.99)) from the sensorimotor projection set (see Eq. (5.97)) are solely dependent on the hypothesis, we have no fixed reference we can compare elements from individual extended sensorimotor neighborhoods to.

As an inspiration we consider the task of molecular docking which as been introduced in Section 2.3.5. The challenge there is to find a spatial configuration for a given molecule that allows it to 'dock' to another one. From a geometric point of view, a molecule can be seen as a flexible structure where rigid segments are connected to each other by rotatable joints. Finding a suitable docking configuration thus corresponds to the task of finding a configuration for these joints that minimizes an energy function which considers several known intra- and intermolecular forces. This interpretation of a 'molecule' can be applied to the sensorimotor chain Ω_t. As established in Section 5.2.1 the sensorimotor chain Ω_t models the geometry- and thus hypothesis-independent history of the interaction between the agent and it's environment. However, applying a hypothesis h_i (see Section 6.2.1) to Ω_t induces a fixed geometric meaning to motor controls as it maps each control to an element of the matrix Lie group $Q^n_{(\mathcal{H},i)}$ as given by $T_{(\cup,i)}$. Thus it assigns a transformation matrix to each control that describes the effect this control has on the agents position and orientation. Applying this geometric interpretation to the entire sensorimotor chain Ω_t, a sequence of memory nodes connected by controls, folds the chain in a hypothesis-specific manner. However, instead of calculating the intersection with another molecule, we aim on folding the sensorimotor chain in such a way, that an intrinsic energy function is minimized.

To establish such a function we consider that folding Ω_t naturally induces hypothesis-specific spatial relations between individual memory nodes, i.e., in some cases two distinct memory nodes are folded in such a way that they are very close together or even on top of each other. These spatial relations between memory nodes are naturally applied to the information associated to them, namely to the elements from $\mathcal{S}$ and $\mathcal{B}$. If the hypothesis is valid, memory nodes that are folded onto each other have similar information associated to them, as they should represent the same configuration in Q^n. In case of an invalid parameterization however, the information associated to these kinds of memory nodes would differ. Our energy function thus would have to relate similarity of two given memory nodes to their proximity. A visualization of four different parameterizations for the same sensorimotor chain Ω_t is given in Fig. 6.1.



(a) Control c_1 is mapped to $\hat{\mathfrak{g}}_1$ (x-translation) and control c_2 is mapped to $\hat{\mathfrak{g}}_2$ (y-translation).

(b) Control c_1 is mapped to $\hat{\mathfrak{g}}_1$ (x-translation), while control c_2 is mapped to a linear combination of $m\hat{\mathfrak{g}}_2$ (x-translation) and $n\hat{\mathfrak{g}}_3$ (x/y-rotation) where $m > n$.

(c) Control c_1 is mapped to $\hat{\mathfrak{g}}_1$ (x-translation), while control c_2 is mapped to a linear combination of $m\hat{\mathfrak{g}}_2$ (x-translation) and $n\hat{\mathfrak{g}}_3$ (x/y-rotation) where $m < n$.

(d) Control c_1 is mapped to $\hat{\mathfrak{g}}_1$ (x-translation) and control c_2 is mapped to $\hat{\mathfrak{g}}_3$ (x/yy-rotation).

Figure 6.1: Visualization of the folded sensorimotor chain Ω_t, for four different parameterizations. Colored spheres indicate configurations that are connected by two different effective controls $c_1 = [1, 0]^T$ and $c_2 = [0, 1]^T$. The color corresponds to the numerical value of the associated, one-dimensional sample. As the relation between samples and configurations $q \in Q^n$ is given by the sample function Π, we assume that spheres with the same color encode a similar configuration, i.e., two configurations that are in each other's close proximity. Finding a folding that causes similarly colored spheres to align, corresponds to finding a valid parameterization of the controls, i.e., a mapping from C to $T_{id}Q^n$. Figures 6.1a to 6.1d display different types of mappings from the two-dimensional control vector $c \in C$ to the generators of $SE(2)$, given as the hypotheses $h_1, \cdots, h_4$. However, only for Fig. 6.1c no differently colored spheres overlap each other, indicating that the corresponding parameterization of Ω_t, i.e., hypothesis h_3 is valid.

However, the problem with a function that only considers discrete points is, that it relies on memory nodes being folded onto each other. Only in these cases, i.e., in cases where memory nodes are sufficiently close together, we can calculate to which extent the associated elements from $\mathcal{S}$ and $\mathcal{B}$ are similar or different. In cases where the induced folding causes the memory

nodes to be scattered, this approach will fail. A solution for this problem is to not only consider memory nodes as distinct points but consider the underlying structure, i.e., the functional relation between configurations and perceivable information.

Considering the sensorimotor map, each extended sensorimotor neighborhood $\Lambda_{\circledcirc}[\omega_i \uplus \Omega_t]$ establishes a local Euclidean reference frame, based on $C_{\circledcirc}$. If the sensorimotor map has been created with a valid hypothesis, this reference frame is the 'projection' of a local patch on the configuration space $\mathcal{Q}^n$ as given by the function $\tau_{\blacktriangleleft}$ (see Eq. (4.38)). Data points from $\Lambda_{\circledcirc}[\omega_i \uplus \Omega_t]$ now link elements from $C_{\circledcirc}$ to elements from $\mathcal{Q}^n$ to which either Π (see Eq. (4.52)) or ϱ (see Eq. (3.10)) is applied. As established in Section 4.2 we assume a 'locally smooth' relation between configurations and perceivable samples. This means that local patches in $\mathcal{Q}^n$ exist in which $\Pi : \mathcal{Q}^n \to \mathcal{S}$ is assumed to be smooth. Mapping these local patches into $C_{\circledcirc} \subseteq C_* \equiv \mathbb{R}^n$ will preserve this property, as $\tau_{\blacktriangleleft}$ is composed of the linear mapping $\tau_{\circlearrowleft}$ and the matrix logarithm. This implies that whatever locally smooth structure exist on a local neighborhood of $\mathcal{Q}^n$ will be preserved by mapping $\mathcal{Q}^n$ into the local Euclidean reference frame $C_{\circledcirc}$ attached to ω_i. For information regarding empty and occupied portions of $\mathcal{Q}^n$, the situation is slightly different as configurations are either empty or occupied as ϱ decomposes $\mathcal{Q}^n$ into the two disjunct sets $\mathcal{Q}^n_\circ$ and $\mathcal{Q}^n_\bullet$. Thus instead of assuming $\varrho : \mathcal{Q}^n \to \mathcal{B}$ to be locally smooth, we instead assume it to be locally constant, i.e., we expect connected patches on $\mathcal{Q}^n$ to exist that are either empty or occupied. Like in the case of samples, this notion of 'locally consistency' will be preserved when mapping them into $C_{\circledcirc} \subseteq C \equiv \mathbb{R}^n$.

We utilize this locally smooth and locally consistent properties to define a notion of intrinsic conflict. However, instead of considering the similarity between individual memory nodes, consider the entire sensorimotor neighborhood, i.e., the local representation of a patch on $\mathcal{Q}^n$. To that end we construct local instances of the sample function and the occupancy function that only apply to the individual extended sensorimotor neighborhoods they are associated to.

The sensorimotor map $\Lambda[\Xi_u]$ is defined as the set of all projection sets $\Lambda_{\circledcirc}[\xi_i \uplus \Omega_t]$ where each $\Lambda_{\circledcirc}[\xi_i \uplus \Omega_t]$ encodes information of a local neighborhood on $\mathcal{Q}^n$ around the configuration $q_{(\xi,i)} \in \mathcal{Q}^n$ associated to the anchor node $\xi_i \in \Xi_u$. As given in Eqs. (5.47) and (5.48) the extended sample set and projection set are subsets of the extended sensorimotor neighborhoods. Thus for any anchor node ξ_i it holds that $\Lambda_{(\mathcal{S},\circledcirc)}[\xi_i \uplus \Omega_t] \subseteq \Lambda_{(\mathcal{S},\circledcirc)}[q_{(\xi,i)}]$ and $\Lambda_{(\mathcal{B},\circledcirc)}[\xi_i \uplus \Omega_t] \subseteq \Lambda_{(\mathcal{B},\circledcirc)}[q_{(\xi,i)}]$. Tuples from $\Lambda_{(\mathcal{S},\circledcirc)}[\xi_i \uplus \Omega_t]$ and $\Lambda_{(\mathcal{B},\circledcirc)}[\xi_i \uplus \Omega_t]$ are given as $\lambda_{(\mathcal{S},j)} = (c_j, s_j)$ and $\lambda_{(\mathcal{B},j)} = (c_j, \beta_j)$ respectively where $c_j \in C_{\circledcirc}$, $s_j \in \mathcal{S}$ and $\beta_j \in \mathcal{B}$. As these tuples represent the relation between controls, samples and obstructions in a local patch, they can be seen as data points, sampled from 'patch-specific' sections of the sample function Π and the occupancy function ϱ respectively. We define the neighborhood specific sample function as

$$\mathcal{S}[\xi_i] : C_{\circledcirc} \to \mathcal{S} \tag{6.13}$$

and the neighborhood specific obstruction function as

$$\mathcal{B}[\xi_i] : C_{\circledcirc} \to \mathcal{B}^*, \tag{6.14}$$

where in Eq. (6.14) we define $\mathcal{B}^* = [0, 1]$ with $\mathcal{B} \subset \mathcal{B}^*$ and relate the elements from $\mathcal{B}$ to $\mathcal{B}^*$ by defining $\beta_\bullet = 0.0$ and $\beta_\circ = 1.0$. Equation (6.13) and Eq. (6.14) model the sample and occupancy

function for a local patch on $\mathcal{Q}^n$, thus characteristics of Π and ϱ being locally smooth and locally consistent will be captured by $\mathcal{S}[\xi_i]$ and $\mathcal{B}[\xi_i]$ as well.

As Π and ϱ are unknown, so are $\mathcal{S}[\xi_i]$ and $\mathcal{B}[\xi_i]$. Consequently, the only information on $\mathcal{S}[\xi_i]$ and $\mathcal{B}[\xi_i]$ is given as data points from $\Lambda_{(\mathcal{S},\circledcirc)}[\xi_i \uplus \Omega_t]$ and $\Lambda_{(\mathcal{B},\circledcirc)}[\xi_i \uplus \Omega_t]$ respectively. Thus, the question we are faced with is: How intrinsically consistent is the set $\Lambda_{\circledcirc}[\xi_i \uplus \Omega_t]$, assuming that all its data points are sampled from the unknown functions $\mathcal{S}[\xi_i]$ and $\mathcal{B}[\xi_i]$? One way to answer this question is to consider multiple instances of the same $\Lambda_{\circledcirc}[\xi_i \uplus \Omega_t]$, where we divide each instance into a different combination of a test and a training set. We treat each instance as an individual regression problem, i.e., train a model based on the training set and evaluate its performance with respect to the test set. This allows us to perform cross-validation [Bishop, 2006, Chapter 1.3] over all instances, where we associate the mean squared error over all instances with the intrinsic conflict. Thus, instead of using cross-validation to determine the performance of the regression algorithm, we use cross-validation to assess the intrinsic disorder of the underlying data, i.e., the underlying function. The algorithm consists of the following steps:

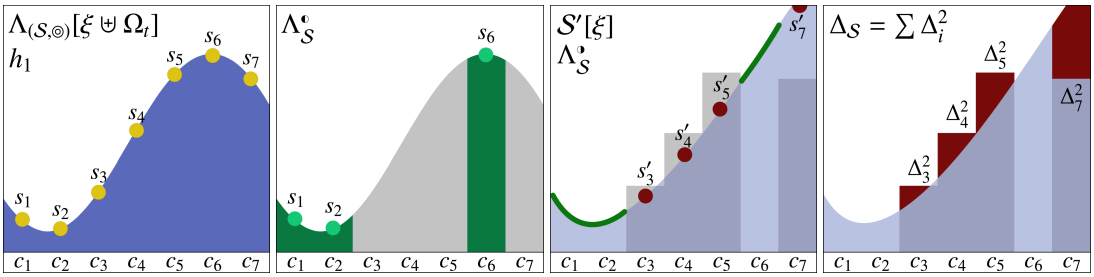
1. Create P non-empty subsets from $\Lambda_{(\mathcal{S},\circledcirc)}[\xi_i \uplus \Omega_t]$ as $\Lambda^{\text{•}}_{(\mathcal{S},1)}, \Lambda^{\text{•}}_{(\mathcal{S},2)}, \cdots, \Lambda^{\text{•}}_{(\mathcal{S},P)}$ where for all $\Lambda_{(\mathcal{S},p)} \subset \Lambda_{(\mathcal{S},\circledcirc)}[\xi_i \uplus \Omega_t]$ it should hold that $|\Lambda^{\text{•}}_{(\mathcal{S},p)}| < |\Lambda_{(\mathcal{S},\circledcirc)}[\xi_i \uplus \Omega_t]|$. Due to the low number of data points in each $\Lambda_{(\mathcal{S},\circledcirc)}[\xi_i \uplus \Omega_t]$, we perform leave-one-out cross-validation in Chapter 7.

2. For each subset $\Lambda^{\text{•}}_{(\mathcal{S},p)}$ approximate $\mathcal{S}[\xi_i]'_p$ based on the data points in $\Lambda^{\text{•}}_{(\mathcal{S},p)}$.

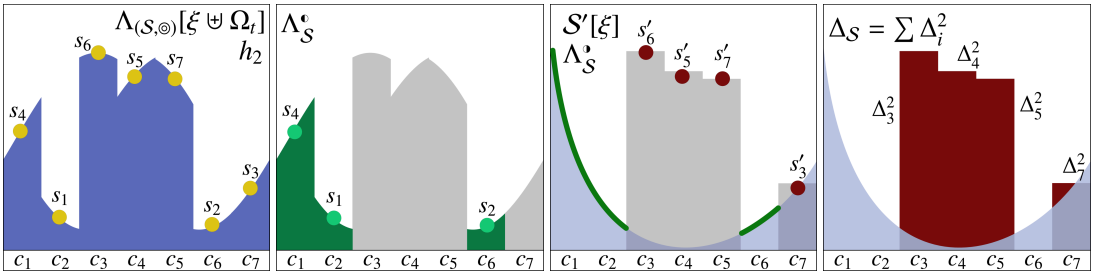
3. Utilize $\mathcal{S}[\xi_i]'_p$ to calculate $\mathcal{S}[\xi_i]'_p(c) = s'$ for every $(c, s) \in \Lambda^{\text{◗}}_{(\mathcal{S},p)}$ where $\Lambda^{\text{◗}}_{(\mathcal{S},p)}$ is the complement of $\Lambda^{\text{•}}_{(\mathcal{S},p)}$ with $\Lambda^{\text{◗}}_{(\mathcal{S},p)} = \Lambda_{(\mathcal{S},\circledcirc)}[\xi_i \uplus \Omega_t] \setminus \Lambda^{\text{•}}_{(\mathcal{S},p)}$.

4. Perform cross-validation by calculating the sum of squared differences between predicted s' and measured s with $\Delta_{(\mathcal{S},p)} = \sum(s_{p,j} - \mathcal{S}[\xi_i]'_p(c_{p,j}))^2 = \sum(s_{p,j} - s'_{p,j})^2$ for all data points in the test set $(c_j, s_j) \in \Lambda^{\text{◗}}_{(\mathcal{S},p)}$, and for all subsets $\Lambda^{\text{◗}}_{(\mathcal{S},1)}, \Lambda^{\text{◗}}_{(\mathcal{S},2)}, \cdots, \Lambda^{\text{◗}}_{(\mathcal{S},P)}$. Divide the sum $\sum_{p=1}^{P} \Delta_{(\mathcal{S},p)}$ by the number of comparisons and associate this value with the intrinsic conflict.

The idea is that, if the hypothesis is valid, data points in $\Lambda_{(\mathcal{S},\circledcirc)}[\xi_i \uplus \Omega_t]$ are created by the function $\mathcal{S}[\xi_i]$ which, as it is a local section of Π, should have the same locally smooth properties. Subset-specific approximations of $\mathcal{S}[\xi_i]$, given as $\mathcal{S}[\xi_i]'_1, \mathcal{S}[\xi_i]'_2, \cdots, \mathcal{S}[\xi_i]'_k$, trained on subsets $\Lambda^{\text{•}}_{(\mathcal{S},1)}, \Lambda^{\text{•}}_{(\mathcal{S},2)}, \cdots, \Lambda^{\text{•}}_{(\mathcal{S},P)}$ should all be similarly able to predict data points from the complements $\Lambda^{\text{◗}}_{(\mathcal{S},1)}, \Lambda^{\text{◗}}_{(\mathcal{S},2)}, \cdots, \Lambda^{\text{◗}}_{(\mathcal{S},P)}$. However, if the hypothesis is invalid, this functional relationship does not exist any more, as the incorrect parametrization of $\Lambda[\Xi_u]$ causes Ω_t to be folded incorrectly, ultimately leading to spatial disorder among the data points. Moreover the degree of disorder will increase, the more the parametrization diverges from the correct one. As in these cases data points were not created by a common function, the predictive performance of subset-specific approximations of $\mathcal{S}[\xi_i]$ will be worse. Examples for a valid and an invalid hypothesis are visualized in Figure 6.2, where we illustrates a single permutation, i.e., a single training and prediction instance, on a valid and invalid projection set, respectively.



(a) The spatial layout of the data points in $\Lambda_{(\mathcal{S},\circledcirc)}$ is induced by the hypothesis h_1. We divide $\Lambda_{(\mathcal{S},\circledcirc)}$ into a training set $\Lambda^{\bullet}_{\mathcal{S}}$ containing the tuples (c_1, s_1), (c_2, s_2), and (c_6, s_6), and a test set, created as its complement with $\Lambda^{\bullet}_{\mathcal{S}} = \Lambda_{\mathcal{S}} \setminus \Lambda^{\bullet}_{\mathcal{S}}$. Approximating the function $\mathcal{S}'[\xi]$ allows us to predict the samples s'_3, s'_4, s'_5, s'_7. The intrinsic conflict of this training set permutation is proportional to the sum of squared differences between s_i and s'_i which is visualized as red shapes.



(b) The spatial layout of the data points in $\Lambda_{(\mathcal{S},\circledcirc)}$ is induced by the hypothesis h_2. We divide $\Lambda_{(\mathcal{S},\circledcirc)}$ into a training set $\Lambda^{\bullet}_{\mathcal{S}}$ containing the tuples (c_4, s_4), (c_1, s_1), and (c_2, s_2), and a test set, created as its complement with $\Lambda^{\bullet}_{\mathcal{S}} = \Lambda_{\mathcal{S}} \setminus \Lambda^{\bullet}_{\mathcal{S}}$. Approximating the function $\mathcal{S}'[\xi]$ allows us to predict the samples s'_6, s'_5, s'_7, s'_3. The intrinsic conflict of this training set permutation is proportional to the sum of squared differences between s_i and s'_i which is visualized as red shapes.

Figure 6.2: Illustration of the intrinsic conflict function for two different hypotheses. Figure 6.2a displays the spatial layout induced by the true hypothesis $h1$. Figure 6.2b, however, displays the spatial layout induced by the hypothesis $h2$, which is incorrect and introduces disorder in the data set. This is indicated by the intrinsic conflict, visualized as red shapes which is considerably larger in Fig. 6.2b. It has to be noted, that we only visualized a single permutation for each case, while the intrinsic conflict is calculated as the average over multiple permutations.

In the next sections we develop two conflict functions, based on two different regression approaches. As we work with subsets of data points from the individual sensorimotor projection sets, as a prerequisite we define P_i non-empty subsets where the p-th subset is given as

$$\Lambda^{\bullet}_{(\mathcal{S},i,p)} = \Lambda^{\bullet}_{(\circledcirc,\mathcal{S},p)}[\xi_i \uplus \Omega_t] \subseteq \Lambda_{(\circledcirc,\mathcal{S})}[\xi_i \uplus \Omega_t] \tag{6.15}$$

and

$$\Lambda^{\bullet}_{(\mathcal{B},i,p)} = \Lambda^{\bullet}_{(\circledcirc,\mathcal{B},p)}[\xi_i \uplus \Omega_t] \subseteq \Lambda_{(\circledcirc,\mathcal{B})}[\xi_i \uplus \Omega_t]. \tag{6.16}$$

We denote the respective complements as

$$\Lambda^{\bullet}_{(\mathcal{S},i,p)} = \Lambda^{\bullet}_{(\circledcirc,\mathcal{S},p)}[\xi_i \uplus \Omega_t] = \Lambda_{(\circledcirc,\mathcal{S})}[\xi_i \uplus \Omega_t] \setminus \Lambda^{\bullet}_{(\circledcirc,\mathcal{S},p)}[\xi_i \uplus \Omega_t] \tag{6.17}$$

and

$$\Lambda^{\blacktriangleright}_{(\mathcal{B},i,p)} = \Lambda^{\blacktriangleright}_{(\circledcirc,\mathcal{B},p)}[\xi_i \uplus \Omega_t] = \Lambda_{(\circledcirc,\mathcal{S})}[\xi_i \uplus \Omega_t] \setminus \Lambda^{\blacktriangleleft}_{(\circledcirc,\mathcal{B},p)}[\xi_i \uplus \Omega_t]. \tag{6.18}$$

Naturally it holds that

$$\begin{aligned} &\Lambda^{\blacktriangleright}_{(\circledcirc,\mathcal{S},p)}[\xi_i \uplus \Omega_t] \cup \Lambda^{\blacktriangleleft}_{(\circledcirc,\mathcal{S},p)}[\xi_i \uplus \Omega_t] = \Lambda_{(\circledcirc,\mathcal{S})}[\xi_i \uplus \Omega_t] \\ &\Lambda^{\blacktriangleright}_{(\circledcirc,\mathcal{S},p)}[\xi_i \uplus \Omega_t] \cap \Lambda^{\blacktriangleleft}_{(\circledcirc,\mathcal{S},p)}[\xi_i \uplus \Omega_t] = \emptyset. \end{aligned} \tag{6.19}$$

and correspondingly

$$\begin{aligned} &\Lambda^{\blacktriangleright}_{(\circledcirc,\mathcal{B},p)}[\xi_i \uplus \Omega_t] \cup \Lambda^{\blacktriangleleft}_{(\circledcirc,\mathcal{B},p)}[\xi_i \uplus \Omega_t] = \Lambda_{(\circledcirc,\mathcal{B})}[\xi_i \uplus \Omega_t] \\ &\Lambda^{\blacktriangleright}_{(\circledcirc,\mathcal{B},p)}[\xi_i \uplus \Omega_t] \cap \Lambda^{\blacktriangleleft}_{(\circledcirc,\mathcal{B},p)}[\xi_i \uplus \Omega_t] = \emptyset. \end{aligned} \tag{6.20}$$

As established in Chapter 1, one of our main objectives is to provide minimal a priori knowledge to the system. With respect to the regression algorithms this corresponds to selecting techniques that make minimal assumptions on the function to be approximated but instead extract as much information from the provided data points. Thus, we look specifically at nonparametric regression algorithms where the underlying concepts are introduced in [Bishop, 2006, Chapter 2] and examples for algorithms are given in [Bishop, 2006, Chapter 6]. As the fundamental assumption is that the similarity between data points increases with decreasing distance on Q^n and thus with decreasing distance in C_*, we utilize linear basis function models in conjunction with Gaussian kernels [Bishop, 2006, Chapters 3.1 and 6.2]. With respect to the extended sensorimotor neighborhoods, a kernel function assigns a positive real number to two elements from $C_\circledcirc$, given by

$$k : C_\circledcirc \times C_\circledcirc \to \mathbb{R}^+. \tag{6.21}$$

The unnormalized Gaussian kernel is given as

$$k(c_j, c_k) = \exp\left(- \frac{d(c_j, c_k)^2}{2\sigma^2} \right) \tag{6.22}$$

where the distance function d on Q^n is given by Eq. (5.111) and the parameter σ denotes the width of the kernel. The aim is to approximate $\mathcal{S}[\xi_i]$ and $\mathcal{B}[\xi_i]$ as a weighted linear combination of $N_\mathcal{S}, N_\mathcal{B} \in \mathbb{N}^*$ kernel functions, as

$$\mathcal{S}[\xi_i]'(c_j) = \sum_{k=1}^{N_\mathcal{S}} w_{(\mathcal{S},k)}\, k_{(\mathcal{S},k)}(c_j, c_k) \quad \text{and} \quad \mathcal{B}[\xi_i]'(c_j) = \sum_{k=1}^{N_\mathcal{B}} w_{(\mathcal{B},k)}\, k_{(\mathcal{B},k)}(c_j, c_k). \tag{6.23}$$

In the following we propose two conflict functions based on two forms of nonparametric regression. The first is a minor modification of the Nadaraya-Watson kernel estimator and the second utilizes a radial basis function network. Though we provide a brief introduction to the underlying concepts, we can only provide a superficial overview within the context of this thesis and thus will refer to further literature.

6.2.2.1 Nadaraya-Watson Kernel Estimator

The Nadaraya-Watson kernel estimator is a nonparametric regression algorithm that estimates an unknown function $f : X \to Y$ as the average of the sum of kernel functions, evaluated at the provided data points. Usually the application of the kernel function is limited by a bandwith parameter H. We slightly modify this algorithm by removing this parameter, thus not applying a hard cut on the area the kernel functions are valid. The reasoning behind this is that the sensorimotor projection sets are bounded by $\epsilon_{\circledcirc}$, the radius of the Euclidean n-ball defining $C_{\circledcirc}$ which corresponds to the radius of the extended sensorimotor neighborhoods. Thus $\epsilon_{\circledcirc}$ can be interpreted as a global implicit bandwith. A brief introduction to the Nadaraya-Watson model can be found in [Bishop, 2006, Chapter 6.3.1]. The Nadaraya-Watson approximation for $\mathcal{S}[\xi_i]$ based on a subset of data points $\Lambda^{\bullet}_{(\mathcal{S},i,p)}$ is given as

$$\Phi_{\mathcal{S}}[\xi_i](c_j, p) = \Big(\sum_{k=1}^{N} k(c_j, c_k) \Big)^{-1} \sum_{k=1}^{N} s_k \, k(c_j, c_k) = \bar{s}_{j,p} \tag{6.24}$$

with $(c_k, s_k) \in \Lambda^{\bullet}_{(\mathcal{S},i,p)}$, $c_j \in C_{\circledcirc}$, and $N = |\Lambda^{\bullet}_{(\mathcal{S},i,p)}|$. The Nadaraya-Watson approximation for $\mathcal{B}[\xi_i]$ based on a subset of data points $\Lambda^{\bullet}_{(\mathcal{B},i,p)}$ is correspondingly given as

$$\Phi_{\mathcal{B}}[\xi_i](c_j, p) = \Big(\sum_{k=1}^{N} k(c_j, c_k) \Big)^{-1} \sum_{k=1}^{N} \beta^*_k \, k_{(\mathcal{B},k)}(c_j, c_k) = \bar{\beta}^*_{j,p} \tag{6.25}$$

with $(c_k, \beta_k) \in \Lambda^{\bullet}_{(\mathcal{B},i,p)}$, $c_j \in C_{\circledcirc}$, $\beta^*_k \in \mathcal{B}^*$, and $N = |\Lambda^{\bullet}_{(\mathcal{B},i,p)}|$. Here $\beta^* \in \mathcal{B}^*$ denotes a real-valued equivalent for $\beta_{\bullet}, \beta_{\circ} \in \mathcal{B}$ (see Eq. (6.14)). Thus, $\beta^*_{\bullet} = 0.0$ and $\beta^*_{\circ} = 1.0$. This representation allows us to interpolate the otherwise symbolic values.

The intrinsic conflict for a given projection set is calculated by applying Eqs. (6.24) and (6.25) to all subsets of $\Lambda_{(\circledcirc,\mathcal{S})}[\xi_i \uplus \Omega_t]$, i.e., $\Lambda_{(\circledcirc,\mathcal{B})}[\xi_i \uplus \Omega_t]$ and their respective complements. In the following, the parameters $P_{(\mathcal{S},i)}, P_{(\mathcal{B},i)} \in \mathbb{N}^*$ denote the number of non-empty subsets of $\Lambda_{(\circledcirc,\mathcal{S})}[\xi_i \uplus \Omega_t]$ and $\Lambda_{(\circledcirc,\mathcal{B})}[\xi_i \uplus \Omega_t]$, respectively. The intrinsic conflict for all samples from $\Lambda_{\circledcirc}[\xi_i \uplus \Omega_t]$ is given as

$$\Delta_{(\mathcal{S},NW)}(\xi_i) = \Big(\sum_{p=1}^{P_{(\mathcal{S},i)}} |\Lambda^{\bullet}_{(\mathcal{S},i,p)}| \Big)^{-1} \sum_{p=1}^{P_{(\mathcal{S},i)}} \Big[\sum_{(c_j,s_j) \in \Lambda^{\bullet}_{(\mathcal{S},i,p)}} (s_j - \bar{s}_{j,p})^2 \Big] \tag{6.26}$$

where $\bar{s}_{j,p}$ refers to Eq. (6.24). Correspondingly the intrinsic conflict for all obstructions from $\Lambda_{\circledcirc}[\xi_i \uplus \Omega_t]$ is given as

$$\Delta_{(\mathcal{B},NW)}(\xi_i) = \Big(\sum_{p=1}^{P_{(\mathcal{B},i)}} |\Lambda^{\bullet}_{(\mathcal{B},i,p)}| \Big)^{-1} \sum_{p=1}^{P_{(\mathcal{B},i)}} \Big[\sum_{(c_j,\beta_j) \in \Lambda^{\bullet}_{(\mathcal{B},i,p)}} (\beta^*_j - \bar{\beta}^*_{j,p})^2 \Big]. \tag{6.27}$$

where $\bar{\beta}^*_{j,p}$ refers to Eq. (6.24). Applying Eq. (6.26) to the entire sensorimotor map yields

$$\Delta_{(\mathcal{S},NW)}(\Lambda[\Xi_u]) = \Big(\sum_{i=1}^{u} \sum_{p=1}^{P_{(\mathcal{S},i)}} |\Lambda^{\bullet}_{(\mathcal{S},i,p)}| \Big)^{-1} \sum_{i=1}^{u} \Big[\sum_{p=1}^{P_{(\mathcal{S},i)}} \Big[\sum_{(c_j,s_j) \in \Lambda^{\bullet}_{(\mathcal{S},i,p)}} (s_j - \bar{s}_{j,p})^2 \Big] \Big]. \tag{6.28}$$

Applying Eq. (6.27) to the entire sensorimotor map yields

$$\Delta_{(\mathcal{B},NW)}(\Lambda[\Xi_u]) = \Big(\sum_{i=1}^{u} \sum_{p=1}^{P_{(\mathcal{B},i)}} |\Lambda_{(\mathcal{B},i,p)}^{\bullet}| \Big)^{-1} \sum_{i=1}^{u} \Bigg[\sum_{p=1}^{P_{(\mathcal{B},i)}} \Bigg[\sum_{(c_j,\beta_j)\in\Lambda_{(\mathcal{B},i,p)}^{\bullet}} (\beta_j^* - \bar{\beta}_{j,p}^*)^2 \Bigg] \Bigg]. \tag{6.29}$$

Combining Eq. (6.28) and Eq. (6.29) yields the intrinsic conflict function

$$\Delta_{NW}(\Lambda[\Xi_u]) = \Delta_{(\mathcal{S},NW)}(\Lambda[\Xi_u]) + \Delta_{(\mathcal{B},NW)}(\Lambda[\Xi_u]). \tag{6.30}$$

6.2.2.2 Radial Basis Function Networks

Radial basis function networks (RBFNs) are nonparametric regression algorithms that, similar to the Nadaraya-Watson kernel estimator discussed in the previous section, utilize kernel functions to approximate a unknown function $f : X \to Y$. The major difference is that radial basis function networks model the output as the sum of $K \in \mathbb{N}^*$ weighted kernel functions $k : C_\circledcirc \times C_\circledcirc \to \mathbb{R}^+$ (see Eq. (6.21)), where their centers $\psi_i \in C_\circledcirc$ are fixed with respect to distinct locations in the input space and are associated with a weight $w_i \in \mathbb{R}$. We define f as

$$f(x) = \sum_{i=1}^{K} w_i \, k(x, \psi_i) \quad \text{with} \quad x \in C_\circledcirc. \tag{6.31}$$

Here $k(x, \psi_i)$ denotes the kernel function associated to the center ψ_i. As both the parameter x and the kernel center ψ_i are elements from $C_\circledcirc$, the kernel function is expressed by using Eqs. (6.21) and (6.22).

To utilize RBFNs for function approximation, a number of design decisions have to be made. In addition to selecting the kernel function, one has to decide on, for example, the number of kernel functions, a strategy for choosing their locations and how to determine their widths. While a brief introduction on RBFNs is given in [Bishop, 2006, Chapter 6.3], for a more comprehensive overview we refer to [Wu et al., 2012]. We will use a straightforward RBFN variant where we utilize a globally fixed kernel width and initialize kernel centers at random data points from subsets of $\Lambda_{(\circledcirc,\mathcal{S})}[\xi_i \uplus \Omega_t]$ and $\Lambda_{(\circledcirc,\mathcal{B})}[\xi_i \uplus \Omega_t]$ respectively. More details on the selection of the kernel width will be given in Section 7.2.5.1.

We denote the RBFN associated to the sensorimotor sample set of a given anchor node $\xi_i \in \Xi_u$ as $\Psi_{\mathcal{S}}[\xi_i]$ and the RBFN associated to the sensorimotor obstruction set as $\Psi_{\mathcal{B}}[\xi_i]$ respectively. The RBFN approximation for $\mathcal{S}[\xi_i]$ trained on the subset $\Lambda_{(\mathcal{S},i,p)}^{\bullet}$ is given as the weighted sum of $K_{(\mathcal{S},i)}$ kernel centers with

$$\Psi_{\mathcal{S}}[\xi_i](c_j, p) = \sum_{k=1}^{K_{(\mathcal{S},i)}} w_{(\mathcal{S},i,k)} \, k(c_j, \psi_{(\mathcal{S},i,k)}) = \bar{s}_{j,p} \tag{6.32}$$

where $w_{(\mathcal{S},i,k)}$ denotes the k-th weight associated to the k-th kernel center $\psi_{(\mathcal{S},i,k)}$. Accordingly, the RBFN approximation for $\mathcal{B}[\xi_i]$, trained on the subset $\Lambda_{(\mathcal{B},i,p)}^{\bullet}$, is given as the weighted sum

of $K_{(\mathcal{B},i)}$ kernel centers with

$$\Psi_{\mathcal{B}}[\xi_i](\beta_j, p) = \sum_{k=1}^{K_{(\mathcal{B},i)}} w_{(\mathcal{B},i,k)}\, k(c_j, \psi_{(\mathcal{B},i,k)}) = \bar{\beta}^*_{j,p}. \tag{6.33}$$

The intrinsic conflict for all samples from $\Lambda_{\circledcirc}[\xi_i \uplus \Omega_t]$ is given as

$$\Delta_{(\mathcal{S},RBFN)}(\xi_i) = \Big(\sum_{p=1}^{P_{(\mathcal{S},i)}} |\Lambda^{\bullet}_{(\mathcal{S},i,p)}| \Big)^{-1} \sum_{p=1}^{P_{(\mathcal{S},i)}} \Big[\sum_{(c_j,s_j)\in\Lambda^{\bullet}_{(\mathcal{S},i,p)}} (\, s_j - \bar{s}_{j,p}\,)^2 \Big] \tag{6.34}$$

where $\bar{s}_{j,p}$ refers to Eq. (6.32). Correspondingly the intrinsic conflict for all obstructions from $\Lambda_{\circledcirc}[\xi_i \uplus \Omega_t]$ is given as

$$\Delta_{(\mathcal{B},RBFN)}(\xi_i) = \Big(\sum_{p=1}^{P_{(\mathcal{B},i)}} |\Lambda^{\bullet}_{(\mathcal{B},i,p)}| \Big)^{-1} \sum_{p=1}^{P_{(\mathcal{B},i)}} \Big[\sum_{(c_j,\beta_j)\in\Lambda^{\bullet}_{(\mathcal{B},i,p)}} (\beta^*_j - \bar{\beta}^*_{j,p}\,)^2 \Big] \tag{6.35}$$

where $\bar{\beta}^*_{j,p}$ refers to Eq. (6.33). Applying Eq. (6.34) to the entire sensorimotor map yields

$$\Delta_{(\mathcal{S},RBFN)}(\Lambda[\Xi_u]) = \Big(\sum_{i=1}^{u} \sum_{p=1}^{P_{(\mathcal{S},i)}} |\Lambda^{\bullet}_{(\mathcal{S},i,p)}| \Big)^{-1} \sum_{i=1}^{u} \Big[\sum_{p=1}^{P_{(\mathcal{S},i)}} \Big[\sum_{(c_j,s_j)\in\Lambda^{\bullet}_{(\mathcal{S},i,p)}} (\, s_j - \bar{s}_{j,p}\,)^2 \Big] \Big]. \tag{6.36}$$

Applying Eq. (6.35) to the entire sensorimotor map yields

$$\Delta_{(\mathcal{B},RBFN)}(\Lambda[\Xi_u]) = \Big(\sum_{i=1}^{u} \sum_{p=1}^{P_{(\mathcal{B},i)}} |\Lambda^{\bullet}_{(\mathcal{B},i,p)}| \Big)^{-1} \sum_{i=1}^{u} \Big[\sum_{p=1}^{P_{(\mathcal{B},i)}} \Big[\sum_{(c_j,\beta_j)\in\Lambda^{\bullet}_{(\mathcal{B},i,p)}} (\beta^*_j - \bar{\beta}^*_{j,p}\,)^2 \Big] \Big]. \tag{6.37}$$

Combining Eq. (6.36) and Eq. (6.37) yields the intrinsic conflict function

$$\Delta_{RBFN}(\Lambda[\Xi_u]) = \Delta_{(\mathcal{S},RBFN)}(\Lambda[\Xi_u]) + \Delta_{(\mathcal{B},RBFN)}(\Lambda[\Xi_u]). \tag{6.38}$$

6.2.2.3 Computational Considerations

The number of computations required to evaluate Eq. (6.30) and Eq. (6.38) depends on two parameters:

The first parameter is the size of the neighborhoods, i.e., the radius of $C_{\circledcirc}$ given as $\epsilon_{\circledcirc}$. During the projection of the sensorimotor chain into the extended neighborhood of a given memory node, data points are discarded if their distance exceeds $\epsilon_{\circledcirc}$ (see Section 5.2.3 and Eqs. (5.62) and (5.63)). Therefore the selection of $\epsilon_{\circledcirc}$ directly influences the number of data points that have to be considered in each projection set. In the limit case with $\epsilon_{\circledcirc} = \infty$ the entire sensorimotor chain would be projected into a single neighborhood, i.e., the number of data points to consider

in Eqs. (6.30) and (6.38) would correspond to the number of samples and obstructions stored in Ω_t.

The second parameter is the number of subsets $P_{(\mathcal{S},i)}$ and $P_{(\mathcal{B},i)}$ each projection set is decomposed into. As both Eqs. (6.30) and (6.38) effectively perform P-fold cross validation (see [Bishop, 2006, Chapter 1.3]), the limit case would be to decompose $\Lambda_{(\circledcirc,\mathcal{S})}[\xi_i \uplus \Omega_t]$ and $\Lambda_{(\circledcirc,\mathcal{B})}[\xi_i \uplus \Omega_t]$ into $P_{(\mathcal{S},i)} = |\Lambda_{(\circledcirc,\mathcal{S})}[\xi_i \uplus \Omega_t]| - 1$ and $P_{(\mathcal{B},i)} = |\Lambda_{(\circledcirc,\mathcal{B})}[\xi_i \uplus \Omega_t]| - 1$ subsets which would correspond to the leave-one-out cross-validation case.

6.2.2.4 Other Approaches

It has to be noted that the two selected methods utilized in Sections 6.2.2.1 and 6.2.2.2 are not the only choices available when constructing a suitable intrinsic conflict function.

One possible alternative would be artificial neural networks (see Section 2.2.1) which can model arbitrary functional relations between input and an outputs based on a set of training data. However, neural networks usually require manual tuning of their parameters, where [Russell and Norvig, 2003, p. 748] states that *"it must be said that a certain amount of twiddling is needed to get the network structure right and achieve convergence to something close to the global optimum in weight space"*. As one of our main objectives is to reduce a priori knowledge, i.e., knowledge provided to the system in advance, having to manually interfere with the discovery-process to provide a suitable network-topology would be counter-intuitive.

Another possible alternative would be Gaussian processes (GPs), probabilistic kernel-based approaches for nonparametric regression (see [Rasmussen and Williams, 2005] for a comprehensive overview). In this approach observations are interpreted as a single sample of a infinity-dimensional multivariate distribution over functions. Gaussian processes model the dependencies between individual samples by utilizing a kernel function which parameterizes the Gaussian process by defining the covariance matrix for said multivariate distribution. Training the model is performed by maximizing the likelihood using an arbitrary optimization algorithm and interpolation is performed by evaluating the multivariate Gaussian distribution at the requested point. In addition to just being able to calculate the mean, they provide the ability to calculate the variance as well, which directly corresponds to the estimated uncertainty of the approximation. As GPs establish a probabilistic model of the data, utilizing them would be beneficial for both bootstrapping and for any high-level planning algorithm. However, they have multiple disadvantages that make them not immediately usable in our scenario.

A major problem is their poor scaling. A stated by [Rasmussen and Williams, 2005, p. 6] an *"issue of Gaussian process prediction methods is that their basic complexity is $O(n^3)$, due to the inversion of the $n \times n$ matrix"* where n denotes the number of data points. Thus utilizing GPs for the intrinsic conflict function would introduce two conflicting goals: on the one hand adding additional data points would improve the expressiveness of the conflict function, on the other hand adding additional points would quickly slow down the system.

Another problem is that while the dimensionality of the input space can be arbitrarily chosen, the standard GP approach allows only for predicting one-dimensional outputs. Though this would still allow us to utilize GPs for modeling empty and free portions of $\mathcal{Q}^n$, modeling the sample function in this way would restrict us to cases where the agent is equipped with only a single sensor, i.e., where the codomain of Π would be one-dimensional. While [Rasmussen

and Williams, 2005, p. 190] suggests *"to model each output variable as independent from the others and treat them separately"* they immediately state that *"this may lose information and be suboptimal"*.

6.3 Bootstrapping Algorithms

In this section we combine the parameterization of the sensorimotor map established in Section 6.2.1 and the intrinsic conflict functions discussed in Section 6.2.2. We propose two algorithms that allow us to identify the sensorimotor map with minimal intrinsic conflict and thus identify the parameterization, i.e., the hypothesis that is best suited to be compatible with the sensorimotor chain.

To account for both Eq. (6.30) and Eq. (6.38) we use the following generic notation for the conflict function, given as

$$\Delta_\diamond : \Lambda[\Xi_u] \rightarrow \mathbb{R}^+ \tag{6.39}$$

with

$$\Delta_\diamond \in \{ \Delta_{NW}, \Delta_{RBFN} \}. \tag{6.40}$$

6.3.1 Bootstrapping as a Classification Problem

In this section we propose an algorithm for bootstrapping where we interprete it as a classification task. Thus, given a sensorimotor chain Ω_t and a set of hypotheses $\mathcal{H}_p$ the task is to identify the hypothesis that matches the true one, i.e., find the true 'class' of the sensorimotor chain. This corresponds to the task of finding the hypothesis $h_{\min} \in \mathcal{H}_p$ with the minimal intrinsic conflict. This hypothesis provides the matrix Lie group and the parameterization of the sensorimotor map that best explains the sensorimotor interaction of the agent with its environment. The algorithm is separated into three distinct functions Classifier::Init(), Classifier::Update(), and Classifier::Classify() which will be described in the following.

6.3.1.1 Classifier Initialization

Algorithm 9 depicts the function Classifier::Init() that initializes the classifier with a set of hypotheses and the initial sample. The algorithm works as follows:

- Line 1: The classifier is initialized with the initial sample s_1 and a non-empty set of hypotheses as defined by Eq. (6.2).
- Line 3: Create a sensorimotor map for each hypothesis in the provided set according to Algorithm 5. The sensorimotor map $\Lambda[\Xi_1]_{(h,i)}$ is associated to the hypothesis h_i and is thus parameterized with the corresponding matrix Lie group $Q^n_{(h,i)}$ and the projection matrices $T_{(\cup,h,i)}$ and $T_{(\cup,h,i)}$).

Bootstrapping Classifier - Init

1: **procedure** CLASSIFIER::INIT($s_1, \mathcal{H}_k$)
2: **for each** h_i **in** $\mathcal{H}_k$ **do**
▷ Initialize sensorimotor map
3: $\Lambda[\Xi_1]_{(h,i)} \leftarrow$ MAP::INIT(s_i, $Q^n_{(h,i)}$, $T_{(\cup,h,i)}$, $T_{(\cup,h,i)}$) ▷ Algorithm 5
4: **end for**
5: **end procedure**

Algorithm 9: Initializing the bootstrapping classifier.

Bootstrapping Classifier - Update

Require: Classifier initialized
1: **procedure** CLASSIFIER::UPDATE($c^{\ominus}_{t-1}, c^{\otimes}_{t-1}, s_t$)
▷ Update sensorimotor map for each hypothesis
2: **for each** h_i **in** $\mathcal{H}_p$ **do**
3: $\Lambda[\Xi_u]_{(h,i)} \leftarrow$ MAP::UPDATE($c^{\ominus}_{t-1}$, $c^{\otimes}_{t-1}$, s_t) ▷ Algorithms 6 and 7
4: **end for**
5: **end procedure**

Algorithm 10: Updating the bootstrapping classifier.

6.3.1.2 Classifier Update

Algorithm 10 depicts the function CLASSIFIER::UPDATE() that updates the sensorimotor map associated to each hypothesis with the latest scheduled and effective controls and the current sample. The algorithm works as follows:

- Line 1: Update the classifier with the latest scheduled control $c^{\ominus}_{t-1}$, the latest effective control $c^{\otimes}_{t-1}$ and the latest sample s_t.
- Line 3: For each hypothesis the corresponding sensorimotor is updated according to Algorithms 6 and 7 by calling MAP::UPDATE().

6.3.1.3 Classification

Algorithm 11 depicts the function CLASSIFIER::CLASSIFY() that evaluates all sensorimotor maps with respect to the conflict function and determines the map with minimal intrinsic conflict. The algorithm works as follows:

- Line 1: Call the classifier and providing the conflict function that should be utilized where according to Eqs. (6.39) and (6.40) $\Delta_\diamond$ corresponds to either Δ_{NW} (see Eq. (6.30)) or Δ_{RBFN} (see Eq. (6.38)).

Bootstrapping Classifier - Classify

1: **procedure** CLASSIFIER::CLASSIFY($\Delta_\diamond$)
2: **for each** h_i **in** $\mathcal{H}_p$ **do**
▷ Calculate conflict
3: $\Delta_{(h,i)} \leftarrow \Delta_\diamond(\Lambda[\Xi_u]_{(h,i)})$ ▷ Eq. (6.30) or Eq. (6.38)
4: **end for**
▷ Return best hypothesis
5: **return** $\arg\min_{h_i \in \mathcal{H}_p} \Delta_{(h,i)}$
6: **end procedure**

Algorithm 11: Utilizing the bootstrapping classifier to identify the hypothesis with the minimal intrinsic conflict from a given set of hypotheses.

- Line 3: Apply the provided conflict function to every sensorimotor map where $\Delta_{(h,i)} \in \mathbb{R}^+$ denotes the conflict associated to the sensorimotor map of hypothesis h_i.
- Line 5: Return the hypothesis h_i for which $\Delta_{(h,i)}$ is minimal.

6.3.2 Bootstrapping as an Optimization Problem

The classification algorithm proposed in the previous section, evaluates the conflict based on a predefined set of hypotheses. In this section we propose another view, where we only provide the matrix Lie group to the system and aim to find the minimal parameterization, effectively treating bootstrapping as an optimization problem. The task of an optimization problem is, given an objective function $f : X \to Y$, to determine $x \in X$ as $\arg\min_{x \in X} f(x)$ or $\arg\max_{x \in X} f(x)$, respectively, depending on the scenario and the objective function. Naturally, the former case is denoted as a minimization, the latter case as a maximization problem. Combining the parameterization of the sensorimotor map given by Eq. (6.9) and the conflict function given by Eq. (6.12) yields the objective function for bootstrapping as

$$(\Delta \circ \Xi \circ \Upsilon)[\Omega_t, Q^n_{\mathcal{H}}] : \mathbb{R}^z \to \mathbb{R}^+. \tag{6.41}$$

This function depends on Ω_t, the hypothesis Lie group $Q^n_{\mathcal{H}}$, a vector from $\mathbb{R}^z$ and the respective instances of the parameterization function $\Upsilon[Q^n_{\mathcal{H}}]$ and the conflict function Δ. As optimization is a wide fields, at this point we refer to [Kelley, 1999] and [Nocedal and Wright, 2006] for an introduction and a good overview of iterative and numerical optimization techniques.

With respect to optimization algorithms, a distinction can be made between approaches that utilize the derivative of the objective function and derivate-free approaches. The latter approaches utilize the objective function 'as it is', i.e., they do not require the derivative to be calculated. The main difficulty in our scenario is that there is no analytic solution for the derivative of $(\Delta \circ \Xi \circ \Upsilon)[\Omega_t, Q^n_{\mathcal{H}}]$, while approximate calculations are computationally expensive. In addition, the input spaces of $\Upsilon[Q^n_{\mathcal{H}}]^\oplus$ and $\Upsilon[Q^n_{\mathcal{H}}]^\ominus$ (Eqs. (6.4) and (6.6)) are continuous, the

sensorimotor map as a graph-structure is not. This means that border cases exist, where slightly changing the parameters for Eqs. (6.4) and (6.6) will change the number of anchor nodes, i.e., will change the number of neighborhoods. As the intrinsic conflict is calculated as the sum of the conflict for all neighborhoods, changing their number will cause discontinuities in the output of $(\Delta \circ \Xi \circ \Upsilon)[\Omega_t, Q^n_{\mathcal{H}}]$ which makes utilizing its derivative impractical.

Thus, we utilize a certain type of derivative-free optimization denoted as particle swarm optimization (PSO). Particle swarm optimization was first introduced in [Kennedy and Eberhart, 1995]. As there exist a large number of variants we refer to [Poli et al., 2007] and [Parsopoulos, 2016] for a good overview. The basic idea is to initialize a number of particles θ. Each particle has a position $x_{(\theta,i)}$ and a velocity $v_{(\theta,i)}$ where $p_{(\theta,i)}, v_{(\theta,i)} \in \mathbb{R}^z$. The algorithm performs multiple iterations where in each iteration initially the objective function is evaluated for each particle at the corresponding position $x_{(\theta,i)}$. Each particle maintains a memory of its individual best position $p^x_{(\theta,i)}$ and the corresponding function value, i.e., the element from $\mathbb{R}^z$ where the function value was minimal. In addition the global best position $g^x_{(\theta,i)}$, i.e., the overall best position over all individual best positions, and the corresponding function value is being kept track of. The velocity of each particle is then modified by applying an attraction towards both the individual best and the global best position where the individual influences are determined by weighting factors. The weighting factor for the individual best is given as w_C, the weighting factor for the global best is w_S. Some approaches introduce an inertia term w_I that keeps the velocity from increasing indefinitely. In each iteration the particle swarm moves through the parameter space, updating its local and global best positions and eventually converging to a (at least local) minimum.

The algorithm presented here is an adaption of the 'Canonical PSO' algorithm given in [Parsopoulos, 2016, Algorithm 1] where slight modifications have been made by integrating the objective function and the mapping from particle positions to the parametrization of the sensorimotor map. Checks to both velocity and parameter space boundaries are implemented to improve the stability of the algorithm. For the velocity this is done by capping the velocity, preventing it to exceed a defined maximum $v_{max} \in \mathbb{R}^+$. For the parameter space this is done by utilizing the 'Reflect-Z' approach described in [Helwig et al., 2013]. In the case of a particle violating the boundaries of the parameter space it's position is 'reflected' and its velocity is set to zero. The stopping criterion has been set to a maximal number of steps $t_{max} \in \mathbb{N}^*$ that were experimentally selected during development.

The algorithm is separated into three distinct functions PSO::Init(), PSO::Update() and PSO::Optimize() which are be described in the following. To account for both the parameterization using $GL(n, \mathbb{R})$ (see Eq. (6.4)) and the parameterization using $SO(n)$ (see Eq. (6.6)) we use the following generic notation, given as

$$\Upsilon[Q^n_{\mathcal{H}}]^\diamond : \Lambda[\Xi_u] \to \mathbb{R}^+ \tag{6.42}$$

with

$$\Upsilon[Q^n_{\mathcal{H}}]^\diamond \in \{\, \Upsilon[Q^n_{\mathcal{H}}]^\oplus,\ \Upsilon[Q^n_{\mathcal{H}}]^\ominus \,\} \tag{6.43}$$

respectively.

6.3.2.1 PSO Initialization

Algorithm 12 depicts the function PSO::INIT(). The algorithm works as follows:

- Line 1: The algorithm requires all parameters for the PSO to be provided. This corresponds to the number of particles n_θ, the number of maximum time steps $t_{\max}$, the parameter space boundaries $x_{min}, x_{max} \in \mathbb{R}^z$ the velocity boundaries $v_{min}, v_{max} \in \mathbb{R}^z$ and the weighting factors w_I, w_C, w_S. Here z is selected according to the choice of the function $\Upsilon[\mathcal{Q}^n_{\mathcal{H}}]^\diamond$. Calling PSO::INIT() requires the initial sample $s_1 \in \mathcal{S}$ to be available. In addition a hypothesis Lie group $\mathcal{Q}^n_{\mathcal{H}}$, a matching parameter function $\Upsilon[\mathcal{Q}^n_{\mathcal{H}}]^\diamond$ and an instance of the parameter function $\Delta_\diamond$ have to be provided.

- Lines 2 to 4: Each of the n_θ particles is initialized with a random position and a random velocity, sampled from a uniform distribution utilizing the parameter space and velocity boundaries respectively.

- Lines 5 and 6: Based on the particle's position $x_{(\theta,i)}$ the particle-specific matrices $T_{(\cup,\theta,i)}$ and $T_{(\cup,\theta,i)}$ are initialized.

- Line 7: The sensorimotor map associated to each particle is initialized with the initial sample s_1 and the particle-specific matrices.

- Lines 8 and 9: The particle's individual best position is initialized with its current position. The particle's individual best evaluation value is initialized as the resulting value of the conflict function, applied to the particle's sensorimotor map.

- Lines 12, 13, 16 and 17: The global best position and evaluation values are initialized and updated with respect to the best particle.

6.3.2.2 PSO Update

Algorithm 13 depicts the function PSO::UPDATE() that updates the sensorimotor map associated to each particle with the latest scheduled and effective controls and the current sample. The algorithm works as follows:

- Line 1: The algorithm requires the PSO to be initialized, i.e., PSO::INIT() must haven been executed beforehand. Calling PSO::UPDATE() requires information on the last scheduled and effective controls $c^{\ominus}_{t-1}, c^{\otimes}_{t-1}$ as well as the latest perceived sample s_t.

- Line 3: Each particle-specific sensorimotor map is updated with the provided information according to Algorithms 6 and 7.

Bootstrapping Particle Swarm Optimization - Init

Require: n_θ initialized
Require: $t_{\max}$ initialized
Require: $x_{min}, x_{max} \in \mathbb{R}^z$ initialized
Require: $v_{min}, v_{max} \in \mathbb{R}^z$ initialized
Require: w_I, w_C, w_S initialized

procedure PSO::INIT($s_1, \mathcal{Q}^n_{\mathcal{H}}, \Upsilon[\mathcal{Q}^n_{\mathcal{H}}]^\diamond, \Delta_\diamond$)
  for each $i \in [1, n_\theta]$ **do**
    ▷ Sample random position and random velocity
    $x_{(\theta,i)} \sim \mathcal{U}(x_{min}, x_{max})$
    $v_{(\theta,i)} \sim \mathcal{U}(v_{min}, v_{max})$
    ▷ Calculate projection matrix from particle position
    $T_{(\cup,\theta,i)} \leftarrow \Upsilon[\mathcal{Q}^n_{\mathcal{H}}]^\diamond(x_{(\theta,i)})$ ▷ Eq. (6.4) or Eq. (6.6)
    $T_{(\cup,\theta,i)} \leftarrow T^{-1}_{(\cup,\theta,i)}$
    ▷ Initialize sensorimotor map
    $\Lambda[\Xi_u]_{(\theta,i)} \leftarrow$ MAP::INIT($s_i, \mathcal{Q}^n_{\mathcal{H}}, T_{(\cup,\theta,i)}, T_{(\cup,\theta,i)}$) ▷ Algorithm 5
    ▷ Initialize individual best position with initial position
    $p^x_{(\theta,i)} \leftarrow x_{(\theta,i)}$
    ▷ Initialize individual best evaluation value
    $p^\Delta_{(\theta,i)} \leftarrow \Delta_\diamond(\Lambda[\Xi_u]_{(\theta,i)})$ ▷ Eq. (6.30) or Eq. (6.38)
    ▷ Create particle
    $\theta_i \leftarrow (x_{(\theta,i)}, v_{(\theta,i)}, p^x_{(\theta,i)}, p^\Delta_{(\theta,i)}, \Lambda[\Xi_u]_{(\theta,i)})$
  end for
  ▷ Initialize global best position and evaluation value
  $g^x \leftarrow 0^z$
  $g^\Delta \leftarrow \infty$
  ▷ Update global best position and evaluation value
  for each θ_i, $i \in [1, n_\theta]$ **do**
    if $p^\Delta_{(\theta,i)} < g^\Delta$ **then**
      $g^x \leftarrow p^x_{(\theta,i)}$
      $g^\Delta \leftarrow p^\Delta_{(\theta,i)}$
    end if
  end for
end procedure

Algorithm 12: Initializing the bootstrapping particle swarm optimization.

Bootstrapping Particle Swarm Optimization - Update

Require: PSO initialized
1: **procedure** PSO::UPDATE($c^{\ominus}_{t-1}, c^{\otimes}_{t-1}, s_t$)
▷ Update sensorimotor map for each particle
2: **for each** θ_i, $i \in [1, n_\theta]$ **do**
3: $\Lambda[\Xi_u]_{(\theta,i)} \leftarrow$ MAP::UPDATE($c^{\ominus}_{t-1}$, $c^{\otimes}_{t-1}$, s_t) ▷ Algorithms 6 and 7
4: **end for**
5: **end procedure**

Algorithm 13: Updating the bootstrapping particle swarm optimization.

6.3.2.3 Optimization

Algorithm 14 depicts the function PSO::OPTIMIZE() that performs the actual optimization by iteratively changing the positions of all particles, rebuilding the associated sensorimotor maps and after a fixed set of iterations return the hypothesis that minimizes the intrinsic conflict. The algorithm works as follows:

- Line 1: The algorithm requires the PSO to be initialized, i.e., PSO::INIT() must haven been executed beforehand.
- Line 3: Perform the main optimization loop t_{max} times.
- Lines 4 and 6 to 8: Each particle's velocity is updated to account for both the individual and the global best. The individual component $(p^x_{(\theta,i)} - x_{(\theta,i)})$ and the global component $(g^x - x_{(\theta,i)})$ are multiplied with the corresponding weighting factors w_C and w_P. Both terms are multiplied with a random factor, sampled from a uniform distribution $\mathcal{U}(0, 1)$. This introduces a random component to the particle's path and thus to the explored area of the parameter space. The function $limit_v()$ prevents $v_{(\theta,i)}$ to exceed $v_{\max}$. If $v_{(\theta,i)} > v_{\max}$ it sets $v_{(\theta,i)}$ to $\|v_{(\theta,i)}\|^{-1} v_{(\theta,i)} v_{\max}$, i.e., normalizes $v_{(\theta,i)}$ and scales it with the maximum velocity. The particle's position is updated with the new velocity. The function $limit_x()$ prevents $x_{(\theta,i)}$ to violate the boundaries of the parameter space. It is implemented as the 'Reflect-Z' approach from [Helwig et al., 2013] that reflects the position of the particle and sets its velocity to zero.
- Lines 9 and 10: Based on the particle's position $x_{(\theta,i)}$ the particle-specific matrices $T_{(\circlearrowleft,\theta,i)}$ and $T_{(\circlearrowright,\theta,i)}$ are updated.
- Line 11: The particle-specific sensorimotor map is rebuilt to account for the changed matrices $T_{(\circlearrowleft,\theta,i)}$ and $T_{(\circlearrowright,\theta,i)}$ utilizing Algorithm 8.
- Lines 12, 14 and 15: If the particle's current evaluation is better than the previous one, the particle's individual best position and evaluation values are updated accordingly.
- Lines 20 and 21: The global best position and evaluation values are updated with respect to the best particle.

- Lines 26 to 29: Finally the hypothesis with the minimal intrinsic conflict is returned.

6.4 Revisiting the Agent Loop

Up until this point we have established the sensorimotor map as a graph-based structure that can be created based on a given sensorimotor chain. We established how to utilized the sensorimotor map to evaluate the intrinsic conflict of a given hypothesis, an interpretation of the agent's sensorimotor interaction with its environment. By using either the classification approach given in Section 6.5.1, or the optimization approach given in Section 6.5.2, the hypothesis that minimizes the conflict can be identified, effectively finding a sensorimotor map that corresponds to the information stored in the sensorimotor chain.

However, we have not yet discussed how these observations, i.e., the sensorimotor chains Ω_t are actually created. Within the intrinsic agent loop given in Algorithm 2 the selection of the next scheduled control $c_t^{\ominus}$ is performed by the DELIBERATE function that, in this algorithm, utilizes the agent's map representation for decision making. However, defining a suitable deliberation functions is an extensive topic on its own that requires considering representing and evaluating objectives, establishing means for path planning and control. As developing a comprehensive navigation algorithm that utilizes the sensorimotor map would exceeded the scope of this thesis, we instead establish a deliberative function that aims on following a certain strategy to explore the unknown environment of the agent. Thus, we propose exploration patterns, predefined sequences of scheduled controls that convey a certain strategy for exploring the environment.

The atomic component of each pattern is given as a sequence of n scheduled controls as elements from $C_{\odot}$ (see Section 4.3.2) with

$$C[n] = (\, c_1^{\ominus},\, c_2^{\ominus},\, \ldots,\, c_n^{\ominus}\,) \tag{6.44}$$

where an exploration pattern is composed of $m \geq 1$ sequences with

$$C[n,m] = (\, C[n]_1,\, C[n]_2,\, \ldots, C[n]_m\,), \tag{6.45}$$

where for the sake of simplicity we assume each to have the same length. Exploration patterns can be either labeled 'simple' or 'returning. Simple patterns describe a random path where each sequence $C[n]_i$ is independent of its preceding sequence $C[n]_{i-1}$. Returning patterns however are composed from an even number of sequences which are constructed in such a way that two successive sequences $(C[n]_1, C[n]_2), (C[n]_3, C[n]_4), \cdots, (C[n]_{m-1}, C[n]_m)$ negate each other. Thus the sequence $C[n]_2$ describes the inverse of the path given by $C[n]_1$. Executing a 'returning' sequence in a non-noisy scenario in an environment with no obstructions, will cause the agent to end at the same location it started from.

These sequences are inspired by the concept of motor babbling (see Section 2.2.3), i.e., the idea is that random sequences of motor controls provide a good foundation for exploring the environment in a scenario where a priori knowledge is scarce. However, instead of selecting completely random sequences, we create them according to certain rules that aim on maximizing the potential intrinsic conflict in the sensorimotor maps and thus accelerate the detection of a

Bootstrapping Particle Swarm Optimization - Optimize

Require: PSO initialized
procedure PSO::OPTIMIZE
  $t \leftarrow 0$
  while $t < t_{max}$ **do**
    for each $\theta_i,\ i \in [1, n_\theta]$ **do**
    ▷ Update particle velocity and apply boundaries
      $v_{(\theta,i)} \leftarrow w_I v_{(\theta,i)} + w_C\, \mathcal{U}(0,1)\,(p^x_{(\theta,i)} - x_{(\theta,i)}) + w_S\, \mathcal{U}(0,1)\,(g^x - x_{(\theta,i)})$
      $v_{(\theta,i)} \leftarrow limit_v(v_{(\theta,i)})$
      ▷ Update particle position and apply boundaries
      $x_{(\theta,i)} \leftarrow x_{(\theta,i)} + v_{(\theta,i)}$
      $x_{(\theta,i)} \leftarrow limit_x(x_{(\theta,i)})$
      ▷ Calculate projection matrix from particle position
      $T_{(\cup,\theta,i)} \leftarrow \Psi[\mathcal{Q}^n_{\mathcal{H}}]^\diamond(x_{(\theta,i)})$
      $T_{(\cup,\theta,i)} \leftarrow T^{-1}_{(\cup,\theta,i)}$
      ▷ Rebuild sensorimotor map and calculate temporary conflict
      $\Lambda[\Xi_u]_{(\theta,i)} \leftarrow$ MAP::REBUILD($\Lambda[\Xi_u]_{(\theta,i)}$, $\mathcal{Q}^n_{(\mathcal{H})}$, $T_{(\cup,\theta,i)}$, $T_{(\cup,\theta,i)}$) ▷ Algorithm 8
      $\Delta_{(\theta,i)} \leftarrow \Delta_\diamond(\Lambda[\Xi_u]_{(\theta,i)})$ ▷ Eq. (6.30) or Eq. (6.38)
      ▷ If necessary update individual best position and evaluation value
      if $\Delta_{(\theta,i)} < p^\Delta_{(\theta,i)}$ **then**
        $p^x_{(\theta,i)} \leftarrow x_{(\theta,i)}$
        $p^\Delta_{(\theta,i)} \leftarrow \Delta_{(\theta,i)}$
      end if
    end for
    ▷ Update global best position and evaluation value
    for each $\theta_i,\ i \in [1, n_\theta]$ **do**
      if $p^\Delta_{(\theta,i)} < g^\Delta$ **then**
        $g^x \leftarrow p^x_{(\theta,i)}$
        $g^\Delta \leftarrow p^\Delta_{(\theta,i)}$
      end if
    end for
    $t \leftarrow t + 1$
  end while
  ▷ Create hypothesis from global best position and return it
  $T_{(\cup,\min)} \leftarrow \Psi[\mathcal{Q}^n_{\mathcal{H}}]^\diamond(g^x)$
  $T_{(\cup,\min)} \leftarrow T^{-1}_{(\cup,\min)}$
  $h_{\min} \leftarrow (\mathcal{Q}^n_{\mathcal{H}}, T_{(\cup,\min)}, T^{-1}_{(\cup,\min)})$
  return $h_{\min}$
end procedure

Algorithm 14: Utilizing the bootstrapping particle swarm optimization to find a parameterization for the sensorimotor map that minimizes the intrinsic conflict.

valid hypothesis. In Chapter 7 we evaluate different kinds of patterns with respect to their utility in creating sensorimotor maps that maximize potential intrinsic conflict.

However, to be able to execute these patterns, we replace the DELIBERATE() function introduced in Section 4.4.3 with a deliberative component. Its behavior depends on whether the provided exploration pattern is a 'simple' or a 'returning' one.

- If $C[n, m]$ is a simple pattern this it simply returns one scheduled control after another. If no more controls are available, it returns a zero control, indicating that the exploration pattern has been executed completely.

- If $C[n, m]$ is a returning pattern, the deliberative component returns one scheduled control after another, like for the 'simple' case. However, it internally keeps track of the ratio between scheduled and effective controls. In case of the agent encountering an obstruction the scheduled and effective controls won't match. In these situations the corresponding 'negative' scheduled control from the matching inverse sequence is adapted to match the effective control. This ensures that the inverse sequence will always match the 'hence' sequence and allows the agent to return to the starting location, even in cases where its path has been obstructed. It has to be noted, that in a scenario in which controls are noisy, the end pose of the agent most likely will differ from its starting pose. As whether or not controls are noisy is unknown to the agent, the deliberative component can not compensate this effect. A visualization for the non-noisy case is depicted in Fig. 6.3.

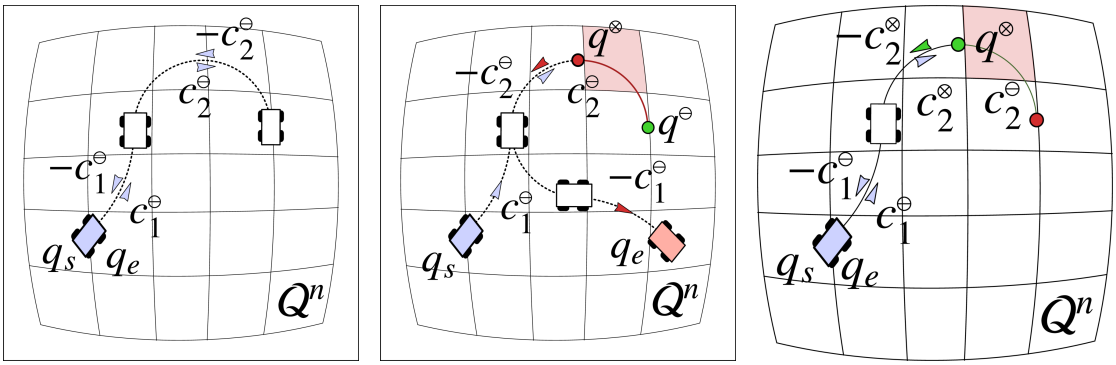
6.5 Bootstrapping using Sensorimotor Maps

In this section we integrate the two bootstrapping algorithms established in Section 6.3 into the agent control loop. The notable difference between those two algorithms is that the classification algorithm can work with arbitrary hypotheses, i.e., arbitrary combinations of a matrix Lie group and the corresponding matrices, as every hypothesis initializes and maintains a separate instance of the sensorimotor map. Within the optimization approach, however, the matrix Lie group has to be the same for all particles. This is due to the fact that all particles are being situated in the same search space. In Section 6.5.1 we adapt the agent loop to incorporate the classification algorithm of Section 6.3.1. In Section 6.5.2 we adapt the agent loop to incorporate the optimization algorithm of Section 6.3.2.

6.5.1 Classification

Algorithm 15 depicts the bootstrapping agent loop that utilizes Algorithms 9 to 11. The algorithm works as follows:

- Line 1: The algorithm requires an exploration pattern, i.e., a sequence of scheduled controls that describes the agent's strategy to explore its environment. In addition a non-empty set of hypotheses and a conflict function has to be provided. The parameter $\epsilon_{\circledcirc}$ has to be initialized as it established the size of the neighborhoods, i.e., the radius of $C_{\circledcirc}$.



(a) The agent starts at the configuration q_s. Executing the sequence $c_1^{\ominus}, c_2^{\ominus}$, followed by its inverse $-c_2^{\ominus}, -c_1^{\ominus}$ results in the end configuration q_e which coincides with q_s.

(b) The agent starts at the configuration q_s. While executing $c_1^{\ominus}$ is possible, executing $c_2^{\ominus}$ is not, as the agent's path is obstructed. The agent collides with the obstacle and thus only partially executes the control, causing the agent's configuration to be $q^{\otimes}$ instead of the intended configuration $q^{\ominus}$. Without incorporating this information, the agent executes $-c_2^{\ominus}, -c_1^{\ominus}$, causing its end configuration q_e to significantly differ from q_s.

(c) The agent starts at the configuration q_s and as depicted in Fig. 6.3b, it executes $c_1^{\ominus}$, a fraction of $c_2^{\ominus}$ and then collides with the obstacle at $q^{\otimes}$. Instead of reversing the scheduled control $c_2^{\ominus}$, the agent reverses $c_2^{\otimes}$ which, per definition (see Eqs. (4.28), (4.29) and (4.59)) corresponds to the fraction of the path actually traveled. After executing $-c_2^{\otimes}, -c_1^{\ominus}$ the agent reaches q_e, which coincides with q_s.

Figure 6.3: Visualization of the difference between an adaptive and a non-adaptive strategy for executing exploration patterns in an environment with obstacles. Depiction of an agent executing the same sequence of controls in multiple non-noisy scenarios. An unobstructed scenario is depicted in Fig. 6.3a. Figure 6.3b depicts a scenario with an obstacle and an agent using a non-adaptive strategy that causes collisions to introduce deviations to the agent's path. Figure 6.3c shows a scenario with an obstacle and an agent that utilizes an adaptive strategy, compensating for the deviations caused from collisions.

- Lines 2 and 3: Scheduled and effective controls for time step t_0 are initialized with the zero control.
- Line 4: The agent perceives the first sample.
- Line 5: The classifier is initialized with the initial sample and the set of hypotheses.
- Lines 6 to 9: The deliberate component provides the next control. To be able to account for obstructions, both the latest scheduled and effective controls have to be provided. If the zero control is returned, the exploration pattern has been completely executed and the agent loop terminates.
- Line 11: The control provided by the deliberate component is executed.
- Lines 12 and 13: The sample at the new configuration is perceived.

- Line 14: The information on the latest control and the new sample is provided to the classifier which in turn updates all internal hypotheses accordingly.
- Lines 16 and 17: The classifier is called after the exploration pattern has been fully executed and the best hypothesis is returned.

6.5.2 Optimization

Algorithm 16 depicts the bootstrapping agent loop that utilizes Algorithms 12 to 14. The algorithm works as follows:

- Line 1: The algorithm requires an exploration pattern, i.e., a sequence of scheduled controls that describes the agent's strategy to explore its environment. In addition the matrix Lie group the optimization will be performed on has to provided, as well as conflict function. The parameter $\epsilon_{\circledcirc}$ has to be initialized as it established the size of the neighborhoods, i.e., the radius of $C_{\circledcirc}$. The parameters of the PSO have to be provided as given by Algorithm 12.
- Lines 2 and 3: Scheduled and effective controls for time step t_0 are initialized with the zero control.
- Line 4: The agent perceives the first sample.
- Line 5: The PSO is initialized with the initial sample and the matrix Lie group.
- Lines 6 to 9: The deliberate component provides the next control. To be able to account for obstructions, both the latest scheduled and effective controls have to be provided. If the zero control is returned, the exploration pattern has been completely executed and the agent loop terminates.
- Line 11: The control provided by the deliberate component is executed.
- Lines 12 and 13: The sample at the new configuration is perceived.
- Line 14: The information on the latest control and the new sample is provided to the PSO which in turn updates all particles accordingly.
- Lines 16 and 17 The PSO optimization algorithm is executed after the exploration pattern has been fully executed and its result is returned.

Self-Discovery Agent Loop (Classification)

Require: non-empty exploration pattern $C[n, m]$ given
Require: non-empty set of hypotheses $\mathcal{H}_k$ given
Require: conflict function $\Delta_\diamond$ given
Require: $\epsilon_\circledcirc$ initialized from $\mathbb{R}^+$
procedure AGENTLOOP :
  ▷ Initialize time index
  $t \leftarrow 1$
  ▷ Initialize scheduled and effective controls for time step t_1
  $c^{\ominus}_{t-1}, c^{\otimes}_{t-1} \leftarrow 0_\odot$
  ▷ Perform initial percept
  $s_t \leftarrow$ PERCEPT() ▷ Eq. (4.56)
  ▷ Initialize classifier
  CLASSIFIER::INIT(s_t, $\mathcal{H}_p$) ▷ Algorithm 9
  ▷ Execute main loop as long as the agent is active
  while *true* **do**
  ▷ Deliberate on next control
    $c^{\ominus}_t \leftarrow$ calculate, based on $C[n, m]$ and ($c^{\ominus}_{t-1}$, $c^{\otimes}_{t-1}$) ▷ Section 6.4
    ▷ Terminate main loop at the end of the exploration pattern
    if $c^{\ominus}_t = 0_\odot$ **then**
      break
    end if
    ▷ Invoke actuator to perform state transition
    $c^{\otimes}_t \leftarrow$ ACT($c^{\ominus}_t$) ▷ Eq. (4.57)
    ▷ Increment time index to indicate state transition
    $t \leftarrow t + 1$
    ▷ Peceive sample at new configuration
    $s_t \leftarrow$ PERCEPT() ▷ Eq. (4.56)
    ▷ Update classifier
    CLASSIFIER::UPDATE($c^{\ominus}_{t-1}$, $c^{\otimes}_{t-1}$, s_t) ▷ Algorithm 10
  end while
  ▷ Calculate intrinsic conflict on all sensorimotor maps
  $h_{best} \leftarrow$ CLASSIFIER::CLASSIFY()
  return h_{best}
end procedure

Algorithm 15: Bootstrapping agent loop, utilizing the classifier approach.

Self-Discovery Agent Loop (Optimization)

Require: non-empty exploration pattern $C[n, m]$ given
Require: matrix Lie group hypothesis $\mathcal{Q}^n_{\mathcal{H}}$ given
Require: conflict function $\Delta_\diamond$ given
Require: PSO parameters initialized (see Algorithm 12)
Require: $\epsilon_\circledcirc$ initialized from $\mathbb{R}^+$

procedure AGENTLOOP
▷ Initialize time index
$t \leftarrow 1$
▷ Initialize scheduled and effective controls for time step t_1
$c^\ominus_{t-1}, c^\ominus_{t-1} \leftarrow 0_\circledcirc$
▷ Perform initial percept
$s_t \leftarrow$ PERCEPT() ▷ Eq. (4.56)
▷ Initialize PSO
PSO::INIT(s_t, $\mathcal{Q}^n_{\mathcal{H}}$) ▷ Algorithm 12
▷ Execute main loop as long as the agent is active
while *true* **do**
▷ Deliberate on next control
$c^\ominus_t \leftarrow$ calculate, based on $C[n, m]$ and ($c^\ominus_{t-1}$, $c^\otimes_{t-1}$) ▷ Section 6.4
▷ Terminate main loop at the end of the exploration pattern
if $c^\ominus_t = 0_\circledcirc$ **then**
break
end if
▷ Invoke actuator to perform state transition
$c^\otimes_t \leftarrow$ ACT($c^\ominus_t$) ▷ Eq. (4.57)
▷ Increment time index to indicate state transition
$t \leftarrow t + 1$
▷ Peceive sample at new configuration
$s_t \leftarrow$ PERCEPT() ▷ Eq. (4.56)
▷ Update PSO
PSO::UPDATE($c^\ominus_{t-1}$, $c^\otimes_{t-1}$, s_t) ▷ Algorithm 13
end while
▷ Calculate intrinsic conflict on all sensorimotor maps
$h_{best} \leftarrow$ PSO::OPTIMIZE()
return h_{best}
end procedure

Algorithm 16: Bootstrapping agent loop, utilizing the particle swarm optimization approach.

6.6 Summary

In this chapter, we utilized the sensorimotor map for bootstrapping, i.e., finding a hypothesis with minimal intrinsic conflict with respect to a given sensorimotor chain. A geometric interpretation of such a hypothesis is that it folds the sensorimotor chain in such a way that it maximizes its intrinsic consistency and thus creates local consistent reference frames that correspond to local patches on the configuration space. The classification and optimization algorithms depicted in Section 6.5 show how this bootstrapping concept can be integrated into a intrinsic agent loop.

Thus, we utilized a simplified deliberate component that executes predefined exploration patterns. The algorithms developed within Chapters 4 to 6 thus should be seen as providing a foundation for higher level concepts like localization, path planning, and more sophisticated deliberation. We discuss on this topic in the outlook in Section 8.2.

7 Evaluation

In this chapter, we describe the evaluation of the bootstrapping algorithms developed in Chapter 6. To that end, we establish the general idea behind the evaluation, depict the setup and the related parameters, and discuss the execution as well as the outcome of the individual evaluation scenarios. The structure is as follows: In Section 7.2, we provide an overview of all relevant parameters and the properties they are associated to. Section 7.2.2 covers all parameters related to the simulation environment, while Section 7.2.3 covers all parameters related to the agent and its sensors. Section 7.2.4 contains information on the different exploration patterns. Section 7.2.5 provides all parameters associated to the sensorimotor map and the parametrization of the bootstrapping algorithms. Section 7.2.6 gives an overview on the hypothesis utilized in the various scenarios. Section 7.3 introduces the visualization used for displaying experiment results and the abbreviations utilized in the tables. Section 7.4 covers the conduction of the experiments. The chapter concludes with a summary in Section 7.5.

7.1 Synopsis

As established in Chapter 1, the aim of this thesis is to establish the sensorimotor map as a domain-independent and parameterizable representation of the entity's configuration space and to utilize it for bootstrapping, i.e., for inferring on the geometric relations between sensory inputs and motor controls. The objective in this chapter is to conduct a number of experiments and evaluate whether the algorithms developed in Section 6.5 are suitable for that task.

As a prerequisite, we define the properties of the simulation system, i.e., the configuration of both the spatial environment and the agent. The environment consists of a cubic, three-dimensionally bounded space through which the agent can move freely, depending on its movement capabilities that are defined by its actuators. The agent is equipped with sensors that allow it to perceive certain properties of the current configuration it is at. These properties are 'light

intensity', 'temperature', 'omnidirectional proximity' to walls, and the 'distance' to the obstacle directly in front of it. In addition to the properties of the simulation system, we introduce a set of four different exploration patterns, i.e., control sequences that cause the agent to follow different paths through the environment.

We then define multiple sets of hypotheses, where, as given in Section 6.2.1, a hypothesis is a combination of a matrix Lie group, defining the assumed topology of the configuration space and the projection matrices, encoding the relation between the agent's motor controls and the effect they have with respect to the manifold's group operations. The hypotheses were created with the aim to systematically create a variety of different projection matrices that cover the parameter space well. We then evaluate the bootstrapping algorithms for classification and optimization, established in Section 6.5 in multiple scenarios.

As a first step, we evaluate whether the conflict functions Eqs. (6.30) and (6.38) developed in Chapter 6 are actually eligible to detect intrinsic conflict. Moreover, we have to evaluate whether the relation between intrinsic conflict and the output of these functions is monotonic, i.e., whether for increasing levels of disorder in the data points the intrinsic conflict is increasing as well. To that end we construct a suitable evaluation scenario in which we can gradually increase the amount of conflict in a set of data points and evaluate the response of the conflict functions. As we vary the kernel parameters of the respective conflict functions, this scenario allows us to identify reasonable parameters to be utilized in the following scenarios.

In the next scenario, we utilize the classification approach (see Section 6.5.1) to compare the effect different exploration patterns have on the classifier's performance. To that end, the four different patterns are evaluated, while the outcome of this scenario allows us to identify an exploration pattern which we can utilize in the following scenarios.

Afterwards, we establish a baseline for the following scenarios with respect to the effect noisy controls and actuators have on the performance of the conflict functions. To that end we calculate the conflict between a given sensorimotor chain and a known ground truth map of the environment for varying levels of actuator and sensor noise.

We then evaluate the classification algorithm with respect to several questions. The first question is how many data points are required to provide a good classification result, i.e., how many controls the agent has to perform to allow for a reasonable assessment of the valid hypothesis. The second question is how the number and size of subsets of data points affects the performance of the conflict functions and thus ultimately the performance of the bootstrapping algorithm. The third question is whether different types of sensors that measure different properties of the environment contribute differently to the classification result. To that end, the classification results of various agents, each equipped with a different combination of sensors, are compared. The fourth question is to which extent noise affects the performance of the bootstrapping algorithm. To that end, we evaluate the classification performance of multiple agents with actuators and sensors subject to varying levels of noise.

We then evaluate the optimization algorithm and investigate to which extent the parameterization of the projection matrices, i.e., the dimension of the search space, affects its performance. In a final scenario we evaluate the optimization algorithm in a setting in which actuators and sensors are subject to noise.

7.2 Evaluation Parameters

In this section we will give an overview of all parameters that are relevant for conducting the experiments described in Section 7.4. Each of these parameters is discussed with respect to the influence it has on certain aspects of the simulation.

All algorithms developed within this thesis have been implemented in C++[1] and have been integrated into a custom software framework. This framework consists of two parts. The first contains the implementation of an agent system that allows one to simulate customizable agents moving through 3-dimensional spatial environments. The second contains the implementation of the sensorimotor map proposed in Chapter 5 and the bootstrapping algorithms proposed in Chapter 6. All experiments described within this chapter are conducted within this framework. The code has been made publicly available for download at:

`https://gitlab.com/rachuy/lsx`

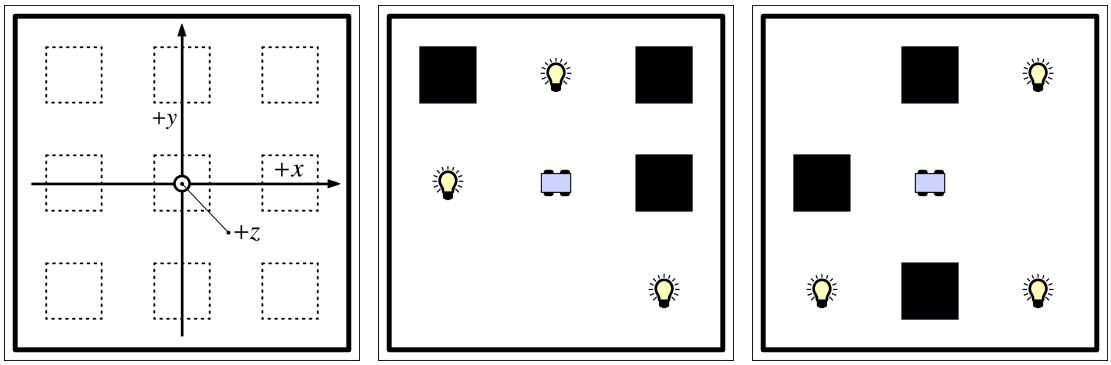
For the sake of clarity and readability some parameters used in the code and their associated values have been renamed or grouped together while maintaining functional equivalence. All experiments are performed on the same system with two Intel Xeon CPU E5-2699 v4 and 256GB of RAM.

7.2.1 Parameter List

This list gives a brief overview over all parameters and provides references to the individual sections in which they are introduced.

Parameter	Description	Reference
`num_iterations`	Number of iterations in each experiment.	Section 7.2.2
`dim_controls`	Dimension of the control vector that defines the degree of freedom of the agent.	Section 7.2.3.1
`noise_actuator`	Standard deviation for the actuator noise function.	Section 7.2.3.1
`enable_sensors`	Sensors the agent is equipped with.	Section 7.2.3.2
`noise_sensors`	Standard deviation for the sensor noise function for all sensors.	Section 7.2.3.2
`agent_stepsize`	Magnitude of each of the agent's controls.	Section 7.2.4
`exploration_pattern`	Exploration pattern type.	Section 7.2.4
`conflict_function`	Intrinsic conflict function used.	Section 7.2.5.1
`conflict_sigma_factor`	Ratio between `agent_stepsize`and the kernel width of the conflict function.	Section 7.2.5.1
`conflict_subsets`	Tuple denoting the number of separate invocations of the conflict function and the fraction of data points utilized in each invocation.	Section 7.2.5.1
`bootstrapping_type`	Bootstrapping algorithm used.	Section 7.2.5.2

[1] Website: `https://isocpp.org`, last accessed 17 November 2019.



(a) Visualization of the simulation environment (top-down view). Positive z-axis points toward the viewer. Dashed squares indicate possible locations for obstructions or light sources. The center square indicates the starting location of the agent.

(b) Visualization of the simulation environment (top-down view) for a possible configuration of obstacles and light sources. Obstacles are visualized as black squares, while light sources are visualized as light bulbs.

(c) Visualization of the simulation environment (top-down view) for a different possible configuration of obstacles and light sources.

Figure 7.1: Conceptual view of the simulation environment and two example configurations.

`pso_num_particles`	Number of PSO particles used.	Section 7.2.5.2
`pso_parameterization`	Used parameterization of projection matrices.	Section 7.2.5.2
`hypothesis_group`	Lie group that defining the configuration space.	Section 7.2.6
`hypothesis_set`	Hypothesis set provided to the agent.	Section 7.2.6

7.2.2 Simulation Environment

The virtual environment the experiments are conducted in is implemented within a custom-built simulation system. The general setup is visualized in Fig. 7.1a while two example configurations are depicted in Figs. 7.1b and 7.1c. The environment consists of a cubic three-dimensional space with a heigh, depth and with of 20 units and is enclosed by impenetrable boundaries, thus restricting the agent to leave this area. At the start of each experiment, i.e., each iteration, the agent is placed in the center of the environment $(0, 0, 0)$ with its front facing the environment's positive x-axis and its up-vector being aligned with the z-axis. As we don't simulate gravity or friction, the agent is not subjected to any external forces which correspond to the assumption of the system being drift-free (see Section 4.2). Thus, in case of simulating movement in $\mathbb{R}^2$, $SO(2)$ or $SE(2)$ the agent 'glides' on an imaginary x/y-plane while in case of simulating movement in $\mathbb{R}^3$, $SO(3)$ or $SE(3)$ the agent is able to freely 'levitate' through space.

In addition to the external boundaries, three solid cubic obstacles with a side length of 2 units are randomly placed in the environment together with three point lights that emit white light. The three cubes as well as the point lights are randomly placed on a 3×3 grid on the x/y-plane of the environment, i.e., the z-coordinate is fixed at $z = 0$. The center grid cell per definition is left empty to avoid creating either a light source or an obstruction directly on the starting location of

the agent. An additional property of the environment is a temperature gradient that is simulated by assigning each $x/y/z$-coordinate the value of a corresponding Perlin noise function. This noise function has been implemented based on the improved noise algorithm given in [Perlin, 2002].

To allow variation in both the simulated environment as well as the exploration patterns, all elements that are dependent on random number generators are initialized based on a global seed which corresponds to the id of each individual experiment run. This strategy ensures 'deterministic randomness', i.e., guarantees that individual iterations within a given experiment differ but at the same time allows them to be repeated. Elements that are dependent on random number generators are the placement of obstacles and light sources, the seed for the Perlin noise, the controls from the individual exploration patterns, the distributions used in the particle swarm optimization and the data point subsets (see Eqs. (6.15) and (6.16)) utilized in Eqs. (6.30) and (6.38) respectively. The relevant parameter is:

- `num_iterations`: Number of iterations within a given experiment. The current iteration ID is used as seed for initializing the random number generators.

7.2.3 Agent Configuration

The agent is implemented as an abstract movable entity that is equipped with a single parameterizable actuator, an arbitrary set of sensors and is situated within the previously defined environment. The shape of the agent is abstracted by a bounding sphere of 0.5 units that is utilized for collision detection. As we do not simulate gravity or dynamic collision responses, the agent has no associated mass.

7.2.3.1 Actuator

The agent is equipped with a single actuator that maps controls, real-valued m-dimensional vectors, to the tangent space of the Lie group that corresponds to the configuration space. Thus with respect to the geometric relation between controls and their translation in the environment, the actuator is parameterized with a hypothesis (see Eq. (6.1)), i.e., a tuple of a matrix Lie group and the corresponding matrices. The degree of freedom of the actuator has to be between one and the dimension of the Lie group, i.e., we allow for holonomic and nonholonomic systems, but not for overactuated systems as established in Section 4.2. Naturally the dimension of the control vector has to be at least one and as the simulation environment corresponds to the three-dimensional Euclidean space, the maximal number of dimensions corresponds to the dimension of the matrix Lie group $SE(3)$ which is six. The actuator can be used with or without noise where in the former case the noise is calculated as zero-mean Gaussian noise, according to Eqs. (4.20) to (4.22), where for the experiments we assume uncorrelated noise with the same variance in each axis. The relevant parameters are:

- `dim_controls`: Dimension of the control vector. It defines the agent's degree of freedom as m in $C_{\odot} \subseteq C \equiv \mathbb{R}^m$. The minimal value is 1 and the maximal value is dependent on the number of generators of the hypothesis matrix Lie group, where the maximal possible value for $SE(3)$ is 6.

- `noise_actuator`: Parameter defining the standard deviation $\sigma_c \in \mathbb{R}^+$ for the actuator noise function, where the actuator-specific covariance matrix is given as $\Sigma^c_{m\times m} = I_{m\times m}\sigma_c^2$.

7.2.3.2 Sensors

Though the previously defined actuator is mandatory, i.e., the agent has to be equipped with exactly one, this restriction does not hold for the sensors. With respect to the simulation system, an arbitrary number of sensors can be attached to the agent at arbitrary poses. However, to limit the total amount of experiments to a reasonable number, we restricted the number of sensors to zero or one for each type of sensor and define the attachment point of each sensor to be the origin of the agent's own reference frame $(0, 0, 0)$, where the x-axis, i.e., the front of the sensor (if applicable) as well as the y- and z-axis of the sensor coincide with those of the agent's reference frame. The output of each individual sensor is implemented to be from the interval $[0, 1]$. For the range sensor, the measurement range of 0 to 20 units is scaled to $[0, 1]$. All sensors can be used with or without noise where in the former case the noise is calculated as zero-mean Gaussian noise, according to Eqs. (4.49) and (4.52), where for the experiments we assume uncorrelated noise with the same variance in each axis.

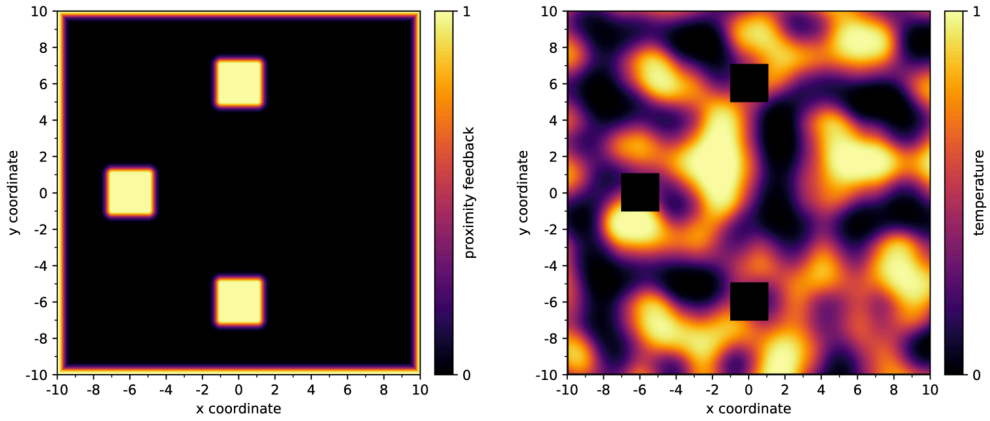
As established in Section 4.3.5 the information available to the agent is a composition of the individual perceptions, thus each sample $s \in \mathcal{S}$ is, from the agent's point of view, just a real-valued vector with a certain dimension. Thus the agent, and moreover the sensorimotor map, has no intrinsic sensor model. The only the conflict functions implicitly create based on the subset of the data points from the sensorimotor projection sets

We implement four different types of sensors with the aim to create sensors with different measurement principles, i.e., sensors whose measurements have different relations to the geometry of the environment:

Light sensor: The light sensor measures the intensity of a light source in its field of view which is modeled as a 180° cone that is aligned with the sensor's x-axis. The measured light intensity is calculated as the dot product between the sensor's view vector and the direction to the light source. Thus, the perceived value depends both on the distance to the light source and its position in the agent's field of view. The feedback is maximal when the agents faces light source directly and decreases when the angle between the view vector and the light source increases.

Temperature sensor: The temperature measures temperature at a given configuration as provided by the Perlin noise function as described in Section 7.2.2. The perception provided by the temperature sensor is only dependent on the position but invariant to the orientation of the agent.

Proximity sensor: The proximity sensor measures the distance between the closes obstacle and the agent and returns feedback that is maximal when the distance reaches zero and exponentially decreases with increasing distance. It is implemented using the fast ray-box intersection algorithm described in [Williams et al., 2005]. The perception provided by the proximity sensor is dependent on the position of the agent and also on the proximity to an obstacle but invariant to the agent's orientation.



(a) Visualization of proximity sensor feedback for different agent positions (x, y). Agent orientation not depicted as proximity sensor feedback is independent from ϕ.

(b) Visualization of temperature sensor feedback for different agent positions (x, y). Agent orientation not depicted as temperature sensor feedback is independent from ϕ.

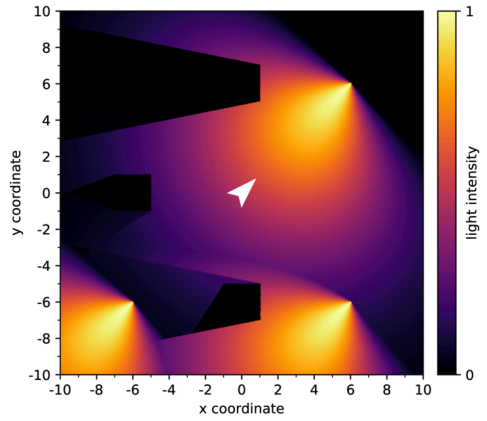
Figure 7.2: Visualization of sensor feedback for the proximity and temperature sensors for the environment configuration given in Fig. 7.1c. Sensor outputs are implemented to be in the interval $[0, 1]$. The outputs have been color coded for better readability.

Range sensor: The range sensor measures the distance to the next obstacle or wall, starting at the agent itself and following a straight line into the direction the agent is facing, i.e., it is modeled as a single laser beam. The measurable distance is capped to a maximum of 20 units in simulation space, i.e., if the distance is greater than 20 units, the sensor will return the maximum output of 1.0. The perception provided by the range sensor is dependent on the pose, i.e., both the position and orientation of the agent.

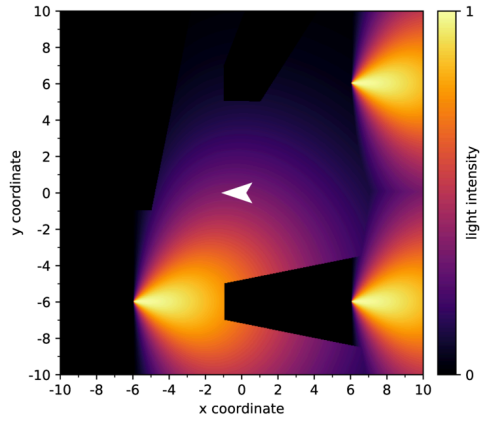
The light sensor provides a continuous, nonlinear feedback that is sensitive to the agent's orientation and may have discontinuities at obstacle borders. The temperature sensor provides a continuous, smooth feedback that is invariant with respect to the agent's orientation. The distance sensor provides continuous linear feedback which is sensitive to the agent's orientation and may have discontinuities at obstacle borders. The proximity sensor provides nonlinear, continuous, orientation-invariant feedback, however only in close proximity to boundaries. A visualization of the feedback for each of these sensors is given in Figs. 7.2 and 7.3. These figures show the sensor feedback for the instantiation of the environment depicted in Fig. 7.1c. Each perception is perceived at a distinct x/y-position while the orientation of the agent remains fixed.

In the majority of experiments, all sensors are activated or deactivated simultaneously. The only exception is Scenario V, where we permute the sensors the agent is equipped with, which will be indicated accordingly. In all experiments all sensors are the subject to the same level of noise. The relevant parameters are:

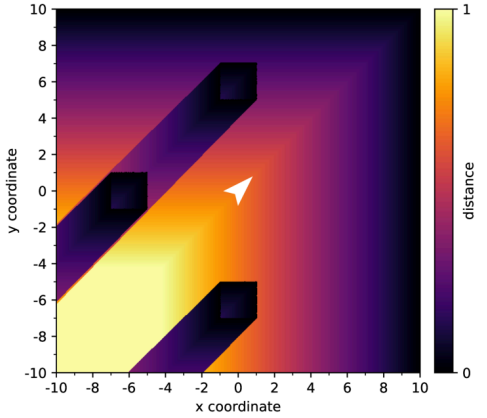
- `enable_sensors`: Boolean indicating whether all sensors are enabled or disabled. Defaukt setting is enabled. Boolean enabling or disabling all sensors Setting properties for



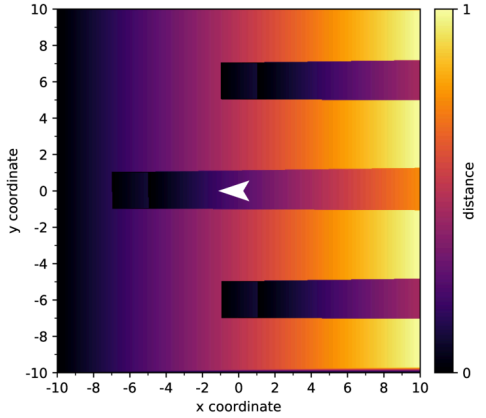
(a) Visualization of the feedback of the light sensor for different agent positions (x, y, ϕ). Agent orientation fixed to $\phi = 45°$.



(b) Visualization of the feedback of the light sensor for different agent positions (x, y, ϕ). Agent orientation fixed to $\phi = 180°$.



(c) Visualization of range sensor feedback for different agent positions (x, y, ϕ). Agent orientation fixed to $\phi = 45°$.



(d) Visualization of range sensor feedback for different agent positions (x, y, ϕ). Agent orientation fixed to $\phi = 180°$.

Figure 7.3: Visualization of sensor feedback for the light and range sensors for the environment configuration given in Fig. 7.1c and for two different orientations of the agent. Sensor outputs are implemented to be in the interval $[0, 1]$. The outputs have been color coded for better readability.

all sensors simultaneously is indicated by <sensors>. Thus, `enable_sensors` activates or deactivates all sensors, while `noise_sensors` sets the standard deviation for all sensors to a fixed value.

- `noise_sensors`: Parameter defining the standard deviation $\sigma_c \in \mathbb{R}^+$ for the sensor noise function, for all sensors. The covariance matrix for each sensor is given as $\Sigma^{p_i}_{r_i \times r_i} = I_{r_i \times r_i} \sigma_c^2$ (see Figs. 7.2 and 7.3) .

7.2.4 Exploration Patterns

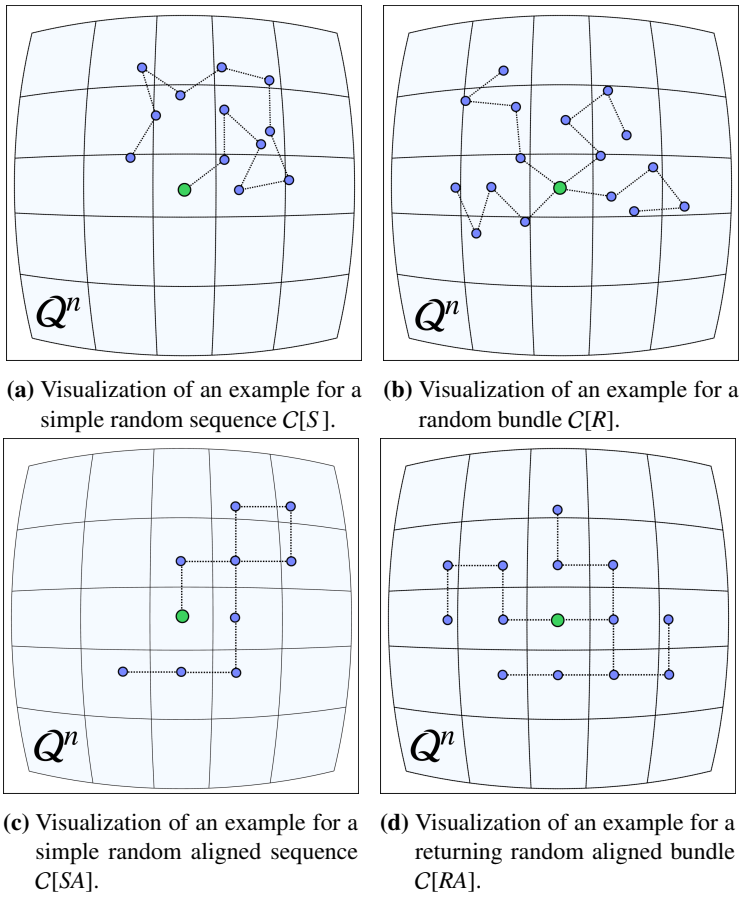
As established in Section 6.4 exploration patterns are predefined sequences of controls the agent subsequently tries to execute. Thus in a way they can be seen as predefined 'plans' or 'strategies' where our aim is to evaluate whether certain strategies generate sensorimotor chains that are better suited for inferring on the hypothesis, i.e., sensorimotor chains that generate maps with more potential intrinsic conflict. As controls are created from subsequently scheduled controls, individual controls have to match the degree of freedom of the agent's control space, i.e., of the agent's actuator. A common parameter of all patterns is the step size which defines the length of each control vector and thus directly influences the distance between two consecutive measurements, i.e., the distance the agent travels on Q^n. Thus, increasing the step size of a given pattern scales this pattern which on the one hand, makes perceptions more scarce and on the other hand, covers more ground. In case of colliding with an obstacle, reverse controls are adaptively modified, based on the proportional feedback (see Section 6.4). The following list defines all patterns used in the experiments where examples for each of these patterns are depicted in Fig. 7.4:

Simple random sequence $C[S]$**:** This pattern consists of a sequence of uniformly distributed random control vectors where only the norm of each control is required to match the step size. This exploration strategy can be related to the concept of motor babbling (see Section 2.2.3 and [Saegusa et al., 2009, Aoki et al., 2016]) where random motor controls are utilized to provide sensorimotor data for training a corresponding representation.

Returning random bundle $C[R]$**:** : This pattern consists of pairs of sequences of random control vectors where as in $C[S]$ the norm of each control is required to match the step size. The second sequence of each pair consists of the reverse controls of the first one. Thus the agent moves to a certain location and, in case of non-noisy controls, returns to the origin from which he moves to another location where this behavior is repeated several times, i.e., the agent performs a number of returning random walks through the environment. This pattern can can be seen as a first step towards incorporating a certain 'strategy' into the pattern. By returning to the origin, i.e., utilizing the inverse controls, the aim is to explore a certain area of the environment, and through subsequent random sequences create a local patch where different 'branches' of the bundle overlap and thus provide a better foundation for calculating the intrinsic conflict then the simple random sequence could.

Simple random aligned sequence $C[SA]$**:** This pattern consists of a sequence of random control vectors where the norm of each control is required to match the step size. In addition each control is required to be axis aligned, i.e., each control can be associated with a distinct dimension of the control vector.

Returning random aligned bundle $C[RA]$**:** This pattern consists of pairs of sequences of random control vectors where as in $C[SA]$ the norm of each control is required to match the step size and the control is required to be axis-aligned. As in $C[R]$ the second sequence of each pair consists of the reverse controls of the first one. This the agent moves to a certain location and, in case of non-noisy controls, returns to the origin from which he moves to



(a) Visualization of an example for a simple random sequence $C[S]$.

(b) Visualization of an example for a random bundle $C[R]$.

(c) Visualization of an example for a simple random aligned sequence $C[SA]$.

(d) Visualization of an example for a returning random aligned bundle $C[RA]$.

Figure 7.4: Examples for different exploration patterns for an actuator with two degrees of freedom. All patterns are created with the same agent step size.

another location where this behavior is repeated several times, i.e., the agent performs a number of returning random walks through the environment. Both $C[SA]$ and this pattern aim on combining the motor babbling approach with a fundamental properties of matrix Lie groups given by the Lie bracket as the commutator of the vector fields given by the group's generators (see Eq. (3.61)). The idea is to exploit that given two hypotheses with differing projection matrices, controls are mapped to potentially non-commuting generators and thus create vastly different projection sets. Thus the idea is to maximize the potential non-commutativity and thus the potential for conflict in the resulting sensorimotor map.

To make these patterns comparable in terms of the number of unique samples created, we adjust the number of steps and sequences accordingly. Both $C[S]$ and $C[SA]$ each contain a single sequence with a total of 1000 control vectors. Both $C[R]$ and $C[RA]$ each contain 20 pairs of 'hence and back' sequences where each of these sequences contains 50 controls. Though the latter two patterns contain a total of 2000 control vectors, only half of them will provide 'new'

information, i.e., new samples as the other half will generate the same samples while 'moving back' through already explored terrain. Thus with respect to the number of novel configurations the agent travels through, all four patterns are designed to be equal, i.e., provide 1000 unique samples. The relevant parameters are:

- `agent_stepsize`: Defines the step size that defines the length of the norm of each individual control in each exploration pattern.
- `exploration_pattern`: Pattern type that the agent should utilize for its exploration, used values are $C[S]$, $C[R]$, $C[SA]$ or $C[RA]$.

7.2.5 Sensorimotor Map and Bootstrapping

During this evaluation we utilize the sensorimotor map algorithm as given in Algorithms 5 to 8. The sensorimotor map is utilized within the classifier algorithm as given in Algorithms 9 to 11 and the optimization algorithm given in Algorithms 12 to 14. Thus the agent executes the agent loops depicted in Algorithm 15 and Algorithm 16 respectively.

The only parameter that has a direct impact on the structure of the sensorimotor map and that is not given by the hypothesis is the size of the individual neighborhoods. It corresponds to $\epsilon_{\circledcirc}$, the radius of the Euclidean n-ball $C_{\circledcirc}$ as given by Eq. (4.10). It influences the creation of the projection sets as given by Eqs. (5.87) and (5.88). We relate $\epsilon_{\circledcirc}$ directly to `agent_stepsize` (see Section 7.2.2) as defining it to be $\epsilon_{\circledcirc} = 1.5 \times$ `agent_stepsize`. Thus the extended sensorimotor neighborhoods and thus the local patches on $\mathcal{Q}^n$ are guaranteed to be slightly larger than the area that can be accessed by the agent in one step, i.e., one execution of a control from $C_{\odot}$. The factor 1.5 is selected with the aim to balance the size of $C_{\circledcirc}$ between 'too small' and 'too' large. The former would cause the sensorimotor map to contain too many neighborhoods with too few data points for the conflict functions to provide useful results. The latter however would cause the sensorimotor map to contain too few neighborhoods with too many data points for the conflict functions to be calculated efficiently, i.e., in a reasonable time.

7.2.5.1 Conflict Function

We will utilize the two conflict functions Δ_{NW} and Δ_{RBFN} (see Eqs. (6.30) and (6.38), respectively), that have been established in Chapter 6. In both conflict functions we utilize the Gaussian kernel as given in Eq. (6.22) where the kernel widths are denoted σ_{NW} and σ_{RBFN} respectively. Selecting reasonable kernel widths is important, as choosing σ either to be too small or to large impairs the performance of the regression algorithms and thus ultimately the expressiveness of the conflict functions. Unfortunately dynamically changing the kernel widths, i.e., adapting them to the properties of the data points contained in each individual sensorimotor neighborhood, might be problematic as this would prevent us from comparing the results of the conflict functions from different neighborhoods, i.e., from different sensorimotor maps. However, assigning a fixed value to σ is difficult as the spatial layout of each set of data points will change, as each parameterization of the sensorimotor chain will cause a different projection Thus, we can not manually tune σ in advance (which also would contradict the idea of providing limited a priori information to the system). We approach this problem by relating σ to `agent_stepsize`,

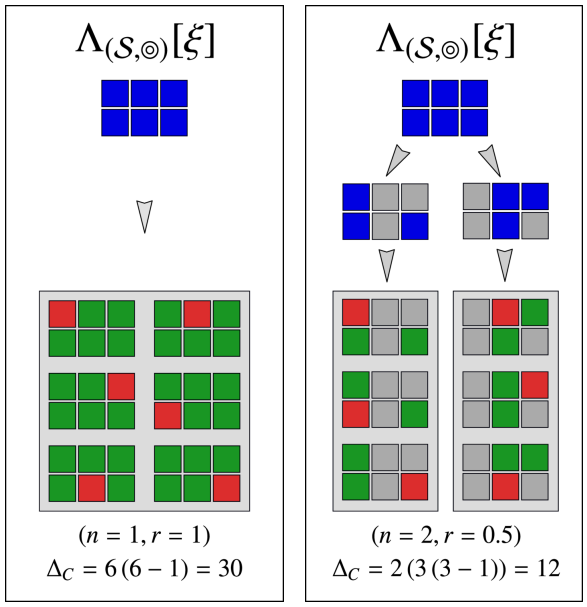
i.e., the distance the agent travels in each individual time step. This parameter defines the distance between two consecutive memory nodes in the sensorimotor chain Ω_t and naturally the distance between their projections in the sensorimotor projection sets $\Lambda_{\circledcirc}[\xi_i \uplus \Omega_t]$. Thus even for parameterizations of the sensorimotor map $\Lambda[\Xi_u]$ that cause oddly shaped spatial distributions of data points in $\Lambda_{\circledcirc}[\xi_i \uplus \Omega_t]$, the distance between consecutive data points will always be constantly given as `agent_stepsize`. By relating the kernel width σ to the agent step size, we directly relate the conflict functions to the geometry of the sensorimotor projection sets $\Lambda_{\circledcirc}[\xi_i \uplus \Omega_t]$. The aim is to chose σ in such a way, that the conflict functions Δ_{NW} and Δ_{RBFN} are equally applicable for all parameterization of the sensorimotor map $\Lambda[\Xi_u]$.

Additional parameters for both Δ_{NW} and Δ_{RBFN} are the number of successive applications of the conflict function on subsets of the data points $n \in \mathbb{N}^*$, and the fraction of data points $r \in (0, 1]$, utilized in each of these subsets. The corresponding parameter is `conflict_subsets`. It allows further tuning the application of the conflict functions. Per default, the number of subsets n is set to one, and the fraction of data points r is set to one as well, accordingly given as the tuple $(1, 1.0)$. For the conflict function Δ_{NW}, this translates to performing leave-one-out cross-validation once (given by $n = 1$), while utilizing the entirety of data points available in each projection set $\Lambda_{\circledcirc}[\xi_i \uplus \Omega_t]$ (given by $r = 1.0$). For the conflict function Δ_{RBFN}, this translates to training and evaluating the RBFN once, utilizing all data points from $\Lambda_{\circledcirc}[\xi_i \uplus \Omega_t]$, separating them into two equally sized training and testing sets.

Modifying `conflict_subsets`, allows us to evaluate whether multiple applications of the conflict functions Δ_{NW} and Δ_{RBFN} on randomly selected, smaller subsets of data points, provide comparable classification and optimization performance, while, at the same time, reducing the computational cost.

Calculating leave-one-out cross-validation on a dataset with N data points, requires $N(N-1)$ pairwise calculations of the distance between data points, which in our case is an expensive calculation, requiring two invocations of the matrix exponential exp and one invocation of the matrix logarithm log. An example for two different parameterizations of `conflict_subsets` for the conflict function Δ_{NW}, and a data set with 6 data points is visualized in Fig. 7.5. Utilizing the full data set with 6 data points, the total number of comparisons is given as $\Delta_C = 6(6-1) = 30$. This corresponds to `conflict_subsets` $= (1, 1.0)$, which is visualized in Figure 7.5a. Utilizing only a subset of the data points, i.e., a fraction $r = 0.5$, yields $\Delta_{C'} = 3(3-1) = 6$ comparisons. To make the total number of invocations of the conflict function comparable, we perform this calculation twice, indicated by selecting $n = 2$. Thus, the total number of comparisons amounts to $\Delta_C = 2(3(3-1)) = 12$. This corresponds to `conflict_subsets` $= (2, 0.5)$, which is visualized in Figure 7.5b. The effect of `conflict_subsets` on the classification performance is evaluated in Section 7.4.5.2. The relevant parameters are:

- `conflict_function`: Defines which conflict function is utilized where used values are: Δ_{NW} and Δ_{RBFN}.
- `conflict_sigma_factor`: Ratio between `agent_stepsize` and the kernel width.
- `conflict_subsets`: Tuple indicating the number of separate invocations of Δ_{NW} or Δ_{RBFN}, and the fraction of data points considered in each of these invocations. Default setting is $(1, 1.0)$.



(a) Number of pairwise distance calculations for the parameter `conflict_subsets` = $(1, 1.0)$ as $\Delta_C = 30$.

(b) Number of pairwise distance calculations for the parameter `conflict_subsets` = $(2, 0.5)$ as $\Delta_C = 12$.

Figure 7.5: Visualization of the effect the selection of `conflict_subsets` has on the computational cost for the conflict function Δ_{NW}. Data set contains 6 data points. Examples given for `conflict_subsets` = $(1, 1.0)$ and `conflict_subsets` = $(2, 0.5)$.

7.2.5.2 Classification and Optimization

We will utilize the classification approach as given in Algorithms 9 to 11 and the optimization approach as given in Algorithms 12 to 14. With respect to the latter we can select between the two parameterization functions $\Upsilon[Q^n_{\mathcal{H}}]^{\ominus}$ and $\Upsilon[Q^n_{\mathcal{H}}]^{\oplus}$, which impacts the size of the parameter space as given in Section 6.2.1. The relevant parameters are:

- `bootstrapping_type`: Defines whether to use the classification or the optimization algorithm. Used values are `classify` for the classifier and `optimize` for the optimization approach.

- `pso_num_particles`: Number of particles to be used in the optimization as given in Algorithms 12 to 14.

- `pso_parameterization`: Indicates whether the optimization approach should utilize $GL(n, \mathbb{R})$ or $SO(n)$ for parameterization the hypotheses which translates to either using Eq. (6.4) or Eq. (6.6). Used values are $\Upsilon[Q^n_{\mathcal{H}}]^{\ominus}$ or $\Upsilon[Q^n_{\mathcal{H}}]^{\oplus}$.

7.2.6 Hypotheses

The effect the actuator has on the agent in terms of movement is defined through a tuple consisting of a Lie group, defining the manifold the agent can move on, and the projection matrices that define the mapping between $C_{\circledcirc}$ and $T_{id}\mathcal{Q}^n$. We will evaluate hypotheses using the groups $SE(2)$ and $SE(3)$ (see Appendices A.2.2 and A.2.2). Elements from $\mathbb{R}^1$, $\mathbb{R}^2$ and $SO(2)$ can be expressed as elements from $SE(2)$, the group of rigid transformations in 2-dimensional Euclidean space. Thus $SE(2)$ covers all scenarios in which a mobile entity moves through a two-dimensional environment. All elements from $SE(2)$ as well as elements from $\mathbb{R}^3$ and $SO(3)$ can be expressed as elements from $SE(3)$, the group of rigid transformations in 3-dimensional Euclidean spacer. Thus $SE(3)$ covers all scenarios in which a mobile entity moves through either a two- or a three-dimensional environment.

For both types of hypotheses we will consider both the holonomic and the nonholonomic case (see Section 3.7) where the number of degrees of freedom of the agent's actuator corresponds to the parameter `dim_controls`(see Section 7.2.2). We distinguish a total of four cases: For hypotheses that utilize $SE(2)$, the holonomic case is given as `dim_controls = 3` and the nonholonomic case as `dim_controls = 2`. For hypotheses that utilize $SE(3)$, the holonomic case is given as `dim_controls = 6` and the nonholonomic case as `dim_controls = 2`, where the latter has been chosen to make the results from both nonholonomic settings comparable.

For both Lie groups we create a set of matrices that map the control vector to the corresponding tangent space and thus either to a single individual generator or a combination of multiple ones. These matrices are linear mappings from $\mathbb{R}^n$ to $\mathbb{R}^n$ (see Section 4.3.3.1) and as established in Section 6.2.1 they can be created by utilizing either the general linear group $GL(n, \mathbb{R})$ or the special orthogonal group $SO(n)$. If we utilize $SO(n)$, each matrix can be visualized as a rotation of the coordinate frame of C^* into $\mathbb{R}^m$, i.e., into the vector space representation of the tangent space $T_{id}\mathcal{Q}^n$. To evaluate the effect that differently scaled control components have, these matrices can be multiplied with a scaling vector that changes the relation between the individual associated tangent space elements.

As there exist arbitrary many matrices (and accordingly many scaling vectors) we have to find a way to (a) systematically create them and (b) limit their number to a feasible amount. The scaling vectors are constructed from vectors $w = [1_1, \ldots, 1_n]^T$ with $n =$ `dim_controls`. Two elements $w_i, w_j \in w$ with $0 \leq i < j \leq n$ are selected and modified by adding or subtracting a factor $f_s = 0.5$. The scaling vectors for $SE(2)$, `dim_controls` $= 3$ and $f_s = 0.5$ are given as

$$
\begin{aligned}
w_1 &= [\,1.0,\ 1.0,\ 1.0\,]^T \\
w_2 &= [\,0.5,\ 1.5,\ 1.0\,]^T \\
w_3 &= [\,1.5,\ 0.5,\ 1.0\,]^T \\
w_4 &= [\,0.5,\ 1.0,\ 1.5\,]^T \\
w_5 &= [\,1.5,\ 1.0,\ 0.5\,]^T \\
w_6 &= [\,1.0,\ 0.5,\ 1.5\,]^T \\
w_7 &= [\,1.0,\ 1.5,\ 0.5\,]^T.
\end{aligned}
\tag{7.1}
$$

These scaling vectors are multiplied to each individual projection matrix. To systematically create them, we utilize their relation to the rotation group $SO(n)$ as introduced in Section 6.2.1. Thus for every generator from $T_{id}SO(n)$, we create a number of matrices by partitioning the full rotation $[0, 2\pi)$ into a number of fixed intervals. As each generator $\hat{\mathfrak{g}} \in T_{id}SO(n)$ represents the rotation in a plane given by two axes in $\mathbb{R}^n$, this approach provides a for every pair of control elements a number of different mappings to generators of Q^n. Selecting $SO(3)$ and an interval of $\frac{\pi}{2}$ yields

$$
\begin{array}{llll}
M_1^{\hat{\mathfrak{g}}_1,\hat{\mathfrak{g}}_2} = \left[\begin{smallmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{smallmatrix}\right] & M_2^{\hat{\mathfrak{g}}_1,\hat{\mathfrak{g}}_2} = \left[\begin{smallmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{smallmatrix}\right] & M_3^{\hat{\mathfrak{g}}_1,\hat{\mathfrak{g}}_2} = \left[\begin{smallmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{smallmatrix}\right] & M_4^{\hat{\mathfrak{g}}_1,\hat{\mathfrak{g}}_2} = \left[\begin{smallmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{smallmatrix}\right] \\
M_1^{\hat{\mathfrak{g}}_1,\hat{\mathfrak{g}}_3} = \left[\begin{smallmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{smallmatrix}\right] & M_2^{\hat{\mathfrak{g}}_1,\hat{\mathfrak{g}}_3} = \left[\begin{smallmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{smallmatrix}\right] & M_3^{\hat{\mathfrak{g}}_1,\hat{\mathfrak{g}}_3} = \left[\begin{smallmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{smallmatrix}\right] & M_4^{\hat{\mathfrak{g}}_1,\hat{\mathfrak{g}}_3} = \left[\begin{smallmatrix} 0 & 0 & -1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{smallmatrix}\right] \\
M_1^{\hat{\mathfrak{g}}_2,\hat{\mathfrak{g}}_3} = \left[\begin{smallmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{smallmatrix}\right] & M_2^{\hat{\mathfrak{g}}_2,\hat{\mathfrak{g}}_3} = \left[\begin{smallmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{smallmatrix}\right] & M_3^{\hat{\mathfrak{g}}_2,\hat{\mathfrak{g}}_3} = \left[\begin{smallmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{smallmatrix}\right] & M_4^{\hat{\mathfrak{g}}_2,\hat{\mathfrak{g}}_3} = \left[\begin{smallmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{smallmatrix}\right]
\end{array}
\tag{7.2}
$$

However, even when considering the three duplicates $M_1^{\hat{\mathfrak{g}}_1,\hat{\mathfrak{g}}_2}$, $M_1^{\hat{\mathfrak{g}}_1,\hat{\mathfrak{g}}_3}$ and $M_1^{\hat{\mathfrak{g}}_2,\hat{\mathfrak{g}}_3}$, this process creates a large number of matrices. Using an interval of $\frac{\pi}{4}$, i.e., steps of 45.0°, would yield 24 matrices and using an interval of $\frac{\pi}{8}$, i.e., steps of 22.5° correspondingly 48 matrices. For $SO(6)$, the Lie group with 15 generators required to parameterize $SE(3)$, even an interval of $\frac{\pi}{4}$ would already create 60 matrices. Additionally, to account for the different scaling vectors, we have to multiply each matrix with each scaling vector. Thus for $SO(n)$, an interval of $\frac{\pi}{4}$ and the 7 scaling vectors defined earlier, the total number of hypotheses to evaluate would correspond to 168 matrices which is impractical. Luckily, we can reduce the total number of matrices by applying two reduction techniques. In the following let $c = [c_1, c_2, c_3]^T$ be a control from $C_{\circledcirc}$ where c_1, c_2, c_3 denote the individual components of c respectively.

- The first filtering technique removes matrices for which a duplicate matrix exists that only differs in the sign of their respective components. As an example consider the matrices

$$
M_1^{\hat{\mathfrak{g}}_2,\hat{\mathfrak{g}}_3} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{and} \quad M_3^{\hat{\mathfrak{g}}_2,\hat{\mathfrak{g}}_3} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix}. \tag{7.3}
$$

  The matrix $M_1^{\hat{\mathfrak{g}}_2,\hat{\mathfrak{g}}_3}$ maps c_1 to $\hat{\mathfrak{g}}_1$, c_2 to $\hat{\mathfrak{g}}_2$ and c_3 to $\hat{\mathfrak{g}}_3$. However $M_3^{\hat{\mathfrak{g}}_2,\hat{\mathfrak{g}}_3}$ does the same, only with the difference that c_2 and c_3 are mapped to $-\hat{\mathfrak{g}}_2$ and $-\hat{\mathfrak{g}}_3$ respectively. Thus two hypotheses utilizing these projection matrices would create the same sensorimotor map, with the only difference that those maps would internally represent a 'mirrored' version of the configuration space. However, the spatial relations between data points and thus the output of the conflict function would be identical.

- The second filtering technique removes matrices that have the same commutative relation between their corresponding tangent space elements. As an example consider the matrices

$$
M_1^{\hat{\mathfrak{g}}_1,\hat{\mathfrak{g}}_2} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{and} \quad M_2^{\hat{\mathfrak{g}}_1,\hat{\mathfrak{g}}_2} = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{7.4}
$$

The matrix $M_1^{\hat{\mathfrak{g}}_1,\hat{\mathfrak{g}}_2}$ maps c_1 to $\hat{\mathfrak{g}}_1$, c_2 to $\hat{\mathfrak{g}}_2$ and c_3 to $\hat{\mathfrak{g}}_3$ where the matrix $M_2^{\hat{\mathfrak{g}}_1,\hat{\mathfrak{g}}_2}$ maps c_1 to -$\hat{\mathfrak{g}}_2$, c_2 to $\hat{\mathfrak{g}}_1$ and c_3 to $\hat{\mathfrak{g}}_3$. Applying the Lie bracket as the matrix commutator (see Section 3.6) it holds that $[\mathfrak{g}_1, \mathfrak{g}_2] = [-\mathfrak{g}_2, \mathfrak{g}_1] = 0_{3\times3}$, i.e., c_1 and c_2 commute (see Eq. (3.76)). Like in the previous case, two hypotheses utilizing these projection matrices would create the same sensorimotor map, with the only difference that those maps would internally represent a 'mirrored' version of the configuration space, in this case 'flipped' along the c_1-axis. The spatial relations between data points and thus the output of the conflict function would be identical in this case too.

For $SE(2)$ in the holonomic case, the number of matrices can be reduced from 84 to 14, 66 to 5, and from 30 to 3 respectively. In the nonholonomic case, the number of matrices can be reduced from 36 to 9, 66 to 9, and 30 to 5, respectively. For $SE(3)$ in the holonomic case, the number of matrices can be reduced from 27360 to 15 and in the nonholonomic case from 660 to 8. We utilized these reduction techniques to create six hypothesis sets that are utilized in conjunction with the classifier algorithm. The superscript $\oplus$ denotes a hypothesis set for the holonomic case, while the superscript $\ominus$ denotes a hypothesis set for the nonholonomic case.

- Hypothesis sets for the holonomic case on $SE(2)$ are given as
  - $\mathrm{H}_1^{\oplus}\mathrm{SE}(2)$ (14 hypotheses) (Appendix C.1) and
  - $\mathrm{H}_2^{\oplus}\mathrm{SE}(2)$ (5 hypotheses) (Appendix C.2).
- Hypothesis sets for the nonholonomic case on $SE(2)$ are given as
  - $\mathrm{H}_1^{\ominus}\mathrm{SE}(2)$ (9 hypotheses) (Appendix C.4) and
  - $\mathrm{H}_2^{\ominus}\mathrm{SE}(2)$ (9 hypotheses) (Appendix C.5).
- The hypothesis set for the holonomic case on $SE(3)$ is given as
  - $\mathrm{H}^{\oplus}\mathrm{SE}(3)$ (15 hypotheses) (Appendix C.7).
- The hypothesis set for the nonholonomic case on $SE(3)$ is given as
  - $\mathrm{H}^{\ominus}\mathrm{SE}(3)$ (8 hypotheses) (Appendix C.8).

We created another two hypothesis sets that are utilized in conjunction with the optimization algorithm.

- The hypothesis set for the holonomic case on $SE(2)$ is given as
  - $\mathrm{H}_3^{\oplus}\mathrm{SE}(2)$ (3 hypotheses) (Appendix C.3).
- The hypothesis set for the nonholonomic case on $SE(2)$ is given as
  - $\mathrm{H}_3^{\ominus}\mathrm{SE}(2)$ (5 hypotheses) (Appendix C.6).

The relevant parameters are:

- `hypothesis_group`: Defines the hypothesis Lie group to be used. Used values are $SE(2)$ and $SE(3)$.
- `hypothesis_set`: Hypothesis set to be used, depending on the hypothesis group.

7.3 Preliminary Definitions

Having defined all parameters and the sets of hypotheses, the next step is to provide information on the individual experiments. We divided the experiments into the number of scenarios where each scenario aims on investigating a certain property of the sensorimotor map and the bootstrapping algorithms respectively. Each scenario is evaluated multiple times, each with a different layout of the simulation environment and a different initialization of the parameters that depend on a random seed. Each individual run with a different seed is called an iteration. As a prerequisite we will describe how scenarios are initialized and executed which covers initialization of the environment and the agents. Afterwards we give a brief overview on the way evaluation results are depicted and the notation and abbreviations that will be used in the tables.

7.3.1 Simulation and Agent Initialization

In each scenario we will utilize a specific set of hypotheses (see Section 7.2.6) and specific types of agents, either using the classification agent loop (see Algorithm 15) or the optimization agent loop (see Algorithm 16). As both algorithms differ in the way they are initialized, i.e., in the way the utilize a provided hypothesis, we will look at each separately. In the following example, we assume to be in iteration $id = 2$, assume to utilize the exploration pattern $C[S]$, and assume the hypothesis set to be $\mathrm{H}_1^{\oplus}\mathrm{SE}(2)$ (Appendix C.1). We assume a holonomic case with a degree of freedom of `dim_controls` $= 3$ and a step size of `agent_stepsize` $= 0.5$. The number of particles for the optimization algorithm is set to be `pso_num_particles` $= 10$. The parameterization of the projection matrices is set to be $SO(3)$, which corresponds to setting `pso_parameterization` $= \Upsilon[Q^n_{\mathcal{H}}]^{\ominus} = \Upsilon[SE(2)]^{\ominus}$. The simulation system is initialized and executed as follows:

1. The simulation environment is initialized. The random number generators are seeded with the id of the current iteration, in this example $id = 2$. Obstacles and light sources are randomly placed. A randomized exploration pattern $C[S]$ is created which is provided to all agents, i.e., all agents will execute the same controls, although the effects of these controls will differ.

2. For each hypothesis $h_i = (SE(2), T_{(\cup,i)}, T_{(\cup,i)})$ (see Eq. (6.1)) from the set $\mathrm{H}_1^{\oplus}\mathrm{SE}(2)$ we create a corresponding agent as $agent_{(h,i)}$. The hypothesis defines the extrinsic parameters, i.e., the true configuration space, and the true mapping between control vectors, Lie group generators and group elements. This extrinsic hypothesis defines the virtual robot hardware and is unknown to the agent. The agent's actuator will be initialized with h_i and scenario-depending noise parameters. For $agent_{(h,i)}$ the hypothesis h_i defines the agent's configuration space to be the matrix Lie group $SE(2)$. The effects of its motor controls are given by the projection matrices $T_{(\cup,i)}$ and $T_{(\cup,i)}$, respectively, that are required by the functions $\tau_{\blacktriangleright}$ and $\tau_{\blacktriangleleft}$ (see Eqs. (4.17) and (4.40)). Sensors are initialized and attached to the agent, where sensor types and noise parameters are scenario-dependent.

3. After the extrinsic parameters have been processed and all agents have been created, each agent is initialized with either a set of hypotheses or a Lie group, depending on whether it utilizes the classification algorithm or the optimization algorithm.

   (a) In scenarios that use classification agents, each agent is initialized with the parameters required for the classification approach Algorithm 15. The exploration pattern is given as $C[S]$. The set of hypotheses is given as $\mathrm{H}_1^{\oplus}\mathrm{SE}(2)$. The conflict function is given as Δ_{NW} or Δ_{RBFN}, depending on the scenario. The neighborhood size is provided as $1.5 \times$ `agent_stepsize` $= 0.75$. For each of the 14 hypotheses from $\mathrm{H}_1^{\oplus}\mathrm{SE}(2)$ the classifier creates an individual sensorimotor map, parameterized with $SE(2)$ and the corresponding projection matrices.

   (b) In scenarios that use optimization agents, each agent is initialized with the parameters required for Algorithm 16. The exploration pattern is given as $C[S]$. The set of hypotheses is given as $\mathrm{H}_1^{\oplus}\mathrm{SE}(2)$. The conflict function is given as Δ_{NW} or Δ_{RBFN}, depending on the scenario. The neighborhood size is provided as $1.5 \times$ `agent_stepsize` $= 0.75$. For each of the 10 particles, a random vector from $\mathbb{R}^3$ is sampled and used to create a projection matrix using $\Upsilon[Q_{\mathcal{H}}^n]^{\ominus}$, i.e., in the case of using $SE(2)$ as the configuration space, $\Upsilon[SE(2)]^{\ominus}$. To each particle, a sensorimotor map is associated, parameterized with $SE(2)$ and the randomized projection matrices.

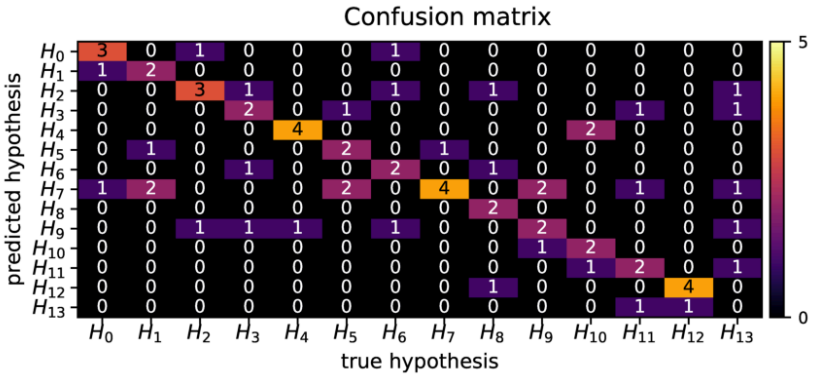
4. The simulation is run, where in each step each agent executes its individual agent loop. The simulation is run until all agents have fully executed their exploration patterns. Final classification and optimization results are extracted, all agents are terminated and the simulation is shut down.

In scenarios that use classification agents, for each of the 14 hypotheses in the set $\mathrm{H}_1^{\oplus}\mathrm{SE}(2)$ an individual agent is created. Each of these agents initializes a classifier that creates 14 sensorimotor maps. Naturally for a given agent, one of these maps will correspond to the true hypothesis, as it is the hypothesis its robot hardware has been initialized with. Likewise, the other 13 hypotheses are false. As for each hypothesis an agent with the corresponding hardware parameterization is created that evaluates all 14 hypotheses, the total number of sensorimotor maps Ξ_u, evaluated in such a scenario, amounts to $|\mathrm{H}_1^{\oplus}\mathrm{SE}(2)|^2 = 14^2 = 196$.

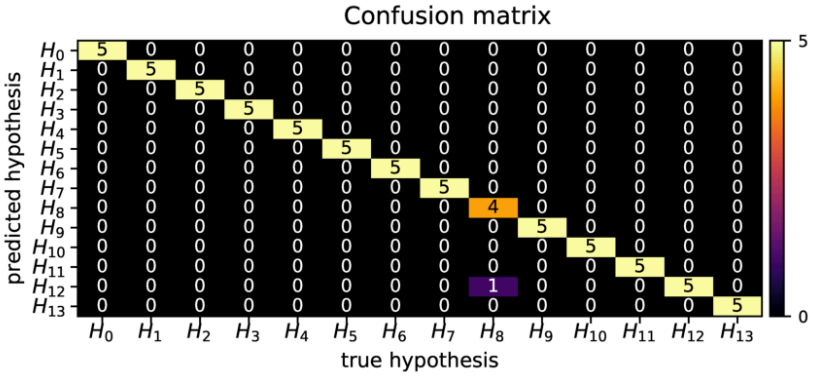
In scenarios that use optimization agents, for each of the 14 hypotheses in the set $\mathrm{H}_1^{\oplus}\mathrm{SE}(2)$ an individual agent is created. Each of these agents initializes an optimizer that creates $n_\theta = 10$ sensorimotor maps with random parameterization. For each cycle of the PSO algorithm, all 10 sensorimotor map are rebuilt and evaluated. For a PSO that runs for $t_{\max} = 50$ cycles, the total number of sensorimotor maps Ξ_u, evaluated in such a scenario, corresponds to $|\mathrm{H}_1^{\oplus}\mathrm{SE}(2)| \times n_\theta \times t_{max} = 14 \times 10 \times 50 = 7000$.

7.3.2 Visualization of Results and Notation

In this section we will introduce the format evaluation results are displayed in. In this regard we discuss two types of figures whose construction we will explain. Additionally we provide a brief explanation of the abbreviations we will utilize in the various tables.



(a) Intermediate confusion matrix after $t = 50$ steps.

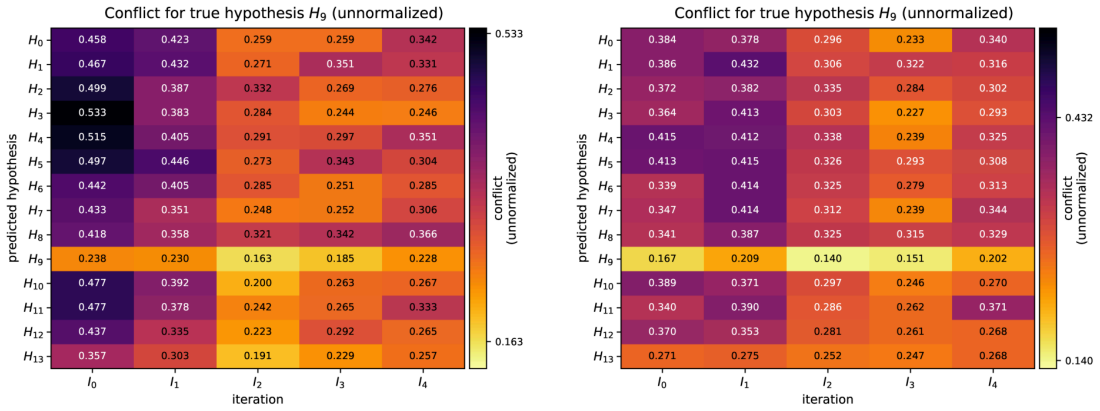


(b) Intermediate confusion matrix after $t = 200$ steps.

Figure 7.6: Confusion matrices for a scenario initialized with hypotheses from $\mathrm{H}_1^{\oplus}\mathrm{SE}(2)$ and 5 iterations. Intermediate results after $t = 50$ and $t = 200$ steps. Each column corresponds to the true hypothesis and represents an agent that has been initialized with a distinct hypothesis. The columns contain the classification results of all 5 iterations. Each row corresponds to a predicted hypothesis. The matrix visualizes the performance of the classifier as each entry corresponds to a combination of true and predicted hypothesis.

In scenarios that use classification agents the results are organized in a confusion matrix. Two examples for such a matrix are depicted in Fig. 7.6. This matrix allows one to asses the overall performance of the classification for all agents as each cell contains information on a distinct combination of 'true' and 'predicted' hypothesis. Figure 7.6a depicts an intermediate case after $t = 50$ steps. The values in column H_9 indicate that hypothesis H_9 has been incorrectly classified as H_7 and H_{10} three times and correctly classified as H_9 two times. The overall performance of the classifier is the sum of diagonal entries divided by the total sum of all entries. In this example this would be $\frac{34}{5\times14} \approx 0.48$. Figure 7.6b depicts the same scenario after $t = 200$ steps. The overall performance of the classifier in this case is $\frac{34}{5\times14} \approx 0.98$.

Even though these confusion matrices enables us to evaluate the performance of the classifier, they only give us a binary view on the results, i.e., evaluate the results of the classifier only as being true or false. As we want to evaluate the effect different parameters have on the conflict function, this binary view is to coarse, i.e., we would like to be able to differentiate the



(a) Partial, intermediate result matrix for agent initialized with H_9 after $t = 200$ steps.

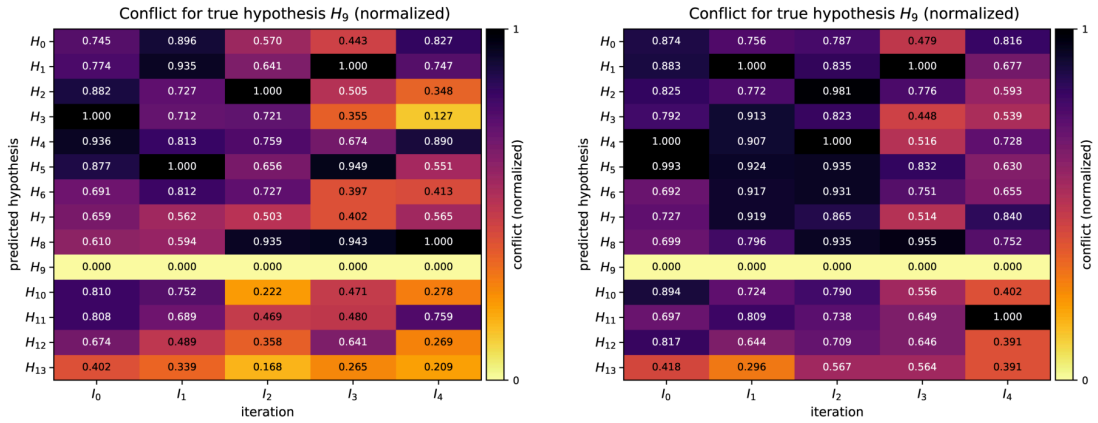
(b) Partial, intermediate result matrix for agent initialized with H_9 after $t = 400$ steps.

Figure 7.7: Partial result matrices for a scenario with five iterations and an agent initialized with hypothesis $H_9 \in \mathrm{H}_1^{\oplus}\mathrm{SE}(2)$. Intermediate results after $t = 200$ and $t = 400$ steps. Each column corresponds to an individual iteration and each row to a predicted hypothesis. Each matrix visualizes the intrinsic conflict of all sensorimotor maps for all predicted hypotheses and all five iterations.

results in terms of a more continuous notion of 'better' and 'worse'. To this end we consider the calculated values of the classifiers directly, i.e., the conflict values associated to each hypothesis, and arrange them in a matrix. Sections of two of these matrices are depicted in figure Fig. 7.7. The matrices display the intermediate results after $t = 200$ and $t = 400$ steps for a scenario initialized with hypotheses from $\mathrm{H}_1^{\oplus}\mathrm{SE}(2)$ and 5 iterations. Each matrix depicts the results for a single agent, initialized with hypothesis H_9. Each row corresponds to a predicted hypothesis and each column corresponds to an individual iteration. Each entry is the result of the conflict function Δ_{NW} or Δ_{RBFN}, applied to he sensorimotor map $\Lambda[\Xi_u]$. In Fig. 7.7a the value 0.238 in row H_9, column I_0 is the return value of $\Delta_{\diamond}(\Lambda[\Xi_u]_{(h,9)})$. This corresponds to the intrinsic conflict of the sensorimotor map, constructed with the parameters of hypothesis H_9 evaluated after having executed 200 controls from the respective exploration pattern.

The problem with this matrix is that entries from different iterations, i.e., entries from different columns can not be compared to each other as each iteration has its own range. Minimum and maximum values in column I_0 in Fig. 7.7b are 0.238 and 0.533 respectively where for I_2 the minimum value is 0.163 and the maximum value 0.332. To make values comparable we perform a column-wise normalization which is depicted in Fig. 7.8. Normalizing each individual column corresponds to normalizing the results of each individual iteration. As this operation is a linear transformation it preserves the order of hypotheses as well as their relative distances.

We combine the partial result matrices of all agents from a given scenario to form a composite result matrix, where two examples for such a matrix are given in Fig. 7.9. Columns in this matrix correspond to both agents and iterations, i.e, for each agent and each iteration an individual, normalized column exists. This matrix allows us to depict the results from an entire scenario in one figure. For Fig. 7.9 this amounts to the results of the conflict calculation applied



(a) Normalized, partial, intermediate result matrix for agent initialized with H_9 after $t = 200$ steps.

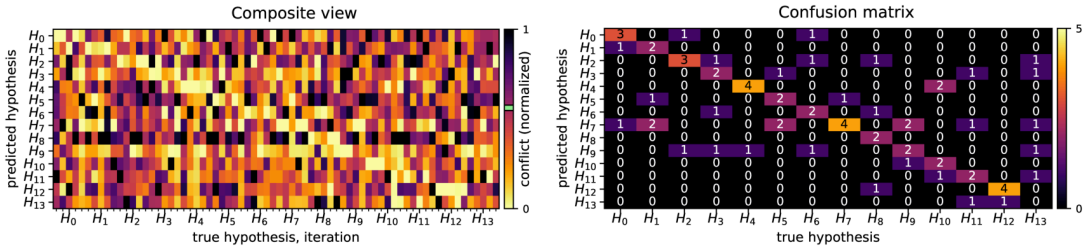
(b) Normalized, partial, intermediate result matrix for agent initialized with H_9 after $t = 400$ steps.

Figure 7.8: Normalized partial result matrices for a scenario with five iterations and an agent initialized with hypothesis $H_9 \in \mathrm{H}_1^{\oplus}\mathrm{SE}(2)$. Intermediate results after $t = 200$ and $t = 400$ steps. Columns are normalized, making individual iterations comparable.

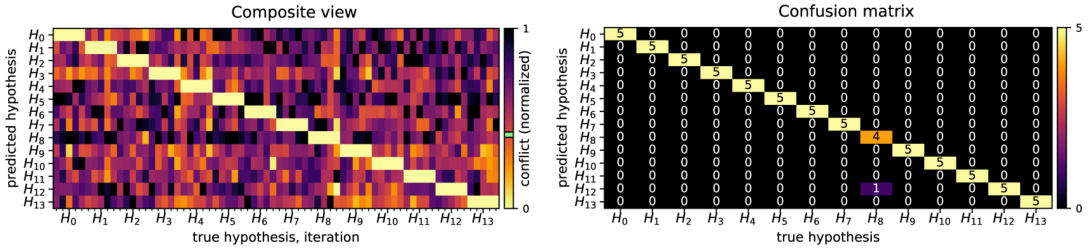
to 980 sensorimotor maps (14 agents with 14 classification hypotheses each, calculated for 5 iterations).

This matrix allows us to visually infer the performance of the intrinsic conflict function. The better, i.e., the more robust the results of the intrinsic conflict function are, the brighter the diagonal of this matrix and the darker the rest of the matrix entries will be. Cases where the conflict function fails to determine the correct hypothesis will result in a matrix where the main diagonal is indistinguishable from the rest of the matrix entries, i.e., a matrix that is overall brighter. The concept of 'brightness' corresponds to the mean of the intrinsic conflict over all agents and iterations.

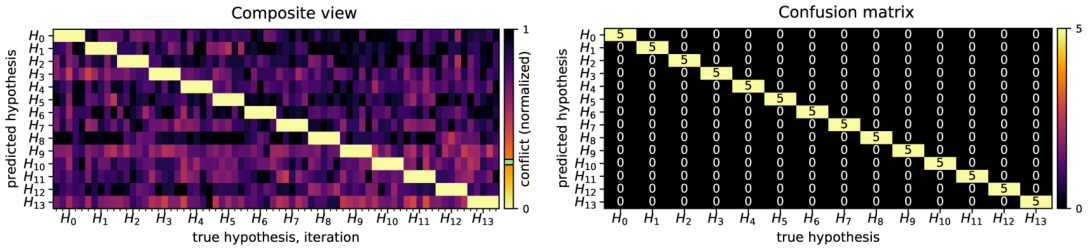
As an example consider the result matrix depicted in Fig. 7.9a, with a mean conflict of 0.562 which is visualized as a green marker at the color bar. To compare the visual impression of the result matrix with the classification result, the corresponding confusion matrix is also depicted in Fig. 7.9a, which visualizes the classification accuracy of 0.49. As a comparison consider Fig. 7.9b and Fig. 7.9c. Though the classification results are similar (0.99 in the former and 1.00 in the latter), the result matrices visually differ as Fig. 7.9b appears brighter than Fig. 7.9c. This corresponds to the mean intrinsic conflict which for Fig. 7.9b is 0.410 and for Fig. 7.9c is only 0.259. Thus, these matrices and the corresponding calculation allows us to perform a more differentiated evaluation of the corresponding scenarios.



(a) Composite intermediate result matrix for $t = 50$ with mean intrinsic conflict of 0.562 and standard deviation 0.301. Corresponding confusion matrix as comparison with accuracy 0.49.



(b) Composite intermediate result matrix for $t = 200$ with mean intrinsic conflict of 0.410 and standard deviation 0.301. Corresponding confusion matrix as comparison with accuracy 0.99.



(c) Composite intermediate result matrix for $t = 1000$ with mean intrinsic conflict of 0.259 and standard deviation 0.242. Corresponding confusion matrix as comparison with accuracy 1.00.

Figure 7.9: Composite result matrices for a scenario with five iterations which is initialized with hypotheses from $\mathrm{H}_1^{\oplus}\mathrm{SE}(2)$. Intermediate results after $t = 50$, $t = 200$ and $t = 1000$ steps. Each block of five columns represents the normalized classification results of an individual agent over the five iterations where each row corresponds to a predicted hypothesis. Mean conflict visualized as green marker in the color bar. Corresponding confusion matrices depicted for comparison.

In addition to these two types of figures we provide additional evaluation results in tables where the following abbreviations are used:

$\Delta[\Xi]/\sum I$	Average intrinsic conflict.
σ_c	Actuator noise.
σ_p	Sensor noise.
p_L, p_P, p_R, p_T	Indicates enabled or disabled **L**ight, **P**roximity, **R**ange or **T**emperature sensor. Enabled sensors denoted [•], disabled sensors denoted [∘].
p_O	Indicates enabled or disabled utilization of occupancy data. Enabled with [•] in all scenarios unless stated otherwise.
$\Lambda[\Xi]$	Number of neighborhoods in the sensorimotor map.
$\mathcal{S}$	Total number of samples stored in the sensorimotor map.
$\mathcal{B}$	Total number of obstacles stored in the sensorimotor map.
T_{CPU} (seconds)	CPU time per hypothesis evaluation.
T_{WALL} (seconds)	Wall clock time per hypothesis evaluation.
$\mathcal{S}/\Lambda[\Xi]$	Average number of samples in each sensorimotor neighborhood.
$\mathcal{B}/\Lambda[\Xi]$	Average number of obstacles in each sensorimotor neighborhood.

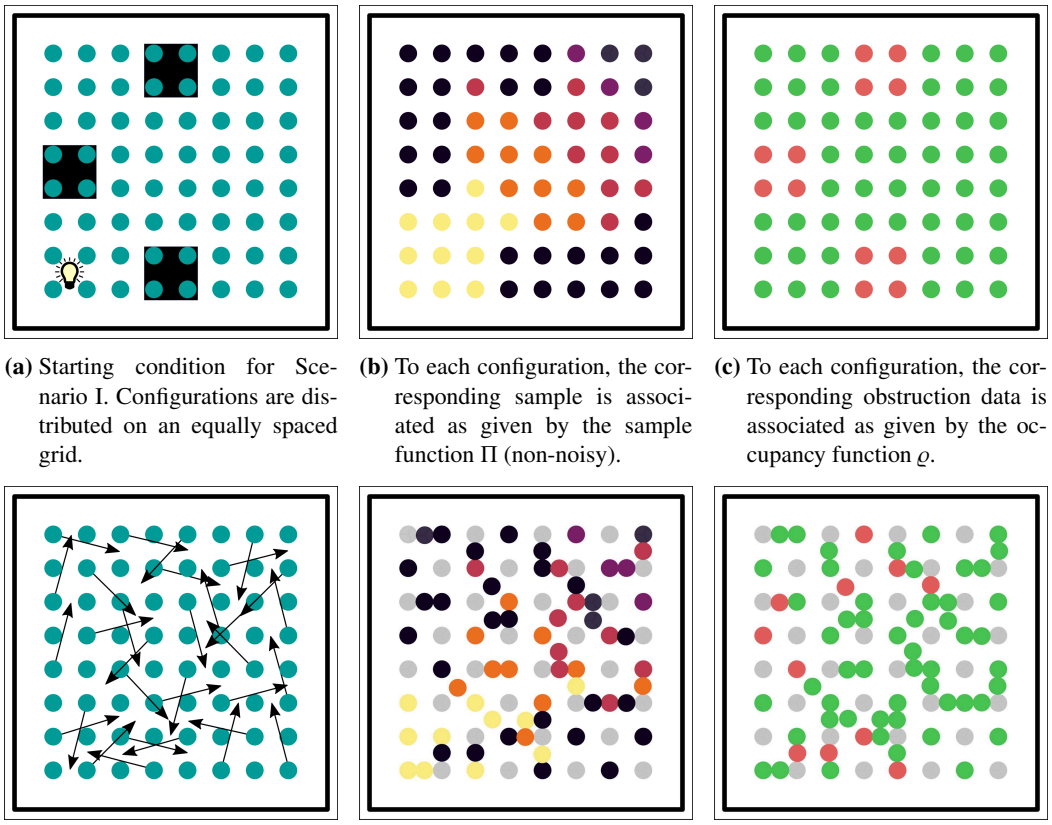
For all but boolean properties the mean μ and the standard deviation σ are provided. In special cases, i.e., certain scenarios, additional symbols are used which are introduced accordingly.

7.4 Experiments

This section covers the evaluation, divided into multiple scenarios. For each scenario we will give a brief overview of its aim and the relevant parameters. Parameters that are not applicable for that scenario are omitted. For each scenario we will discuss the results, the implication and if applicable the impact on following scenarios. Most relevant figures and tables will be depicted in each section while further material can be found in Appendix D.

7.4.1 Scenario I : Conflict Function Evaluation

In this scenario we evaluate whether the conflict functions Δ_{NW} (see Eq. (6.30)) and Δ_{RBFN} (see Eq. (6.38)) are eligible to detect the intrinsic conflict, i.e., are able to detect disorder in a given sensorimotor projection set, where the return value of these functions should correspond to the degree of disorder. In addition this experiment aims on identifying a suitable value for `conflict_sigma_factor`, i.e., a suitable relation between `agent_stepsize` and the kernel widths σ_{NW} and σ_{RBFN}. To that end we initialize an environment and generate sample and obstruction information for configurations on a x/y-grid. This yields a set of tuples $((x, y), s, \beta^*)$ with $s \in \mathcal{S}$ and $\beta^* \in [0, 1]$. To every other data point we assign a randomly chosen unit vector which indicates the direction of displacement, where the setup is depicted in Fig. 7.10. We then gradually move the points along these displacement vectors, i.e., over the course of 20 steps increase their total displacement from 0.0 to a maximum of `agent_stepsize`. For each step the distance between the original location and displaced location increases. Thus we gradually add an increasing amount of disorder to the set of data points as we do not change the sample



(a) Starting condition for Scenario I. Configurations are distributed on an equally spaced grid.

(b) To each configuration, the corresponding sample is associated as given by the sample function Π (non-noisy).

(c) To each configuration, the corresponding obstruction data is associated as given by the occupancy function ϱ.

(d) Assigned to every other configuration is a random unit vector, denoting the displacement direction.

(e) Visualization of the maximal displacement of every other configuration and the associated sample.

(f) Visualization of the maximal displacement of every other configuration and the associated obstruction.

Figure 7.10: Setup for Scenario I, conflict function evaluation. Sample and obstruction measurements for points distributed on a regular grid are generated. To every other data point a displacement vector is associated. Over the course of 20 steps, each of these points is gradually displaced which increases the degree of disorder in the set of data points.

num_iterations	10
dim_controls	–
noise_actuator	0.0
enable_sensors	true
noise_sensors	0.0
agent_stepsize	0.5, 1.0 and 1.5
exploration_pattern	–
conflict_function	Δ_{NW} and Δ_{RBFN}
conflict_sigma_factor	0.1 to 2.0 with a step size of 0.1
conflict_subsets	(1, 1.0)
bootstrapping_type	–
pso_num_particles	–
pso_parameterization	–
hypothesis_group	–
hypothesis_set	–

Table 7.2: Parameters for Scenario I, conflict function evaluation.

and obstruction information but only their (x, y) component. This process is depicted for the final step for samples and obstruction data in Figs. 7.10e and 7.10f, respectively. In each step we evaluate the data points, i.e., the set of all samples and obstructions with respect to both conflict functions.

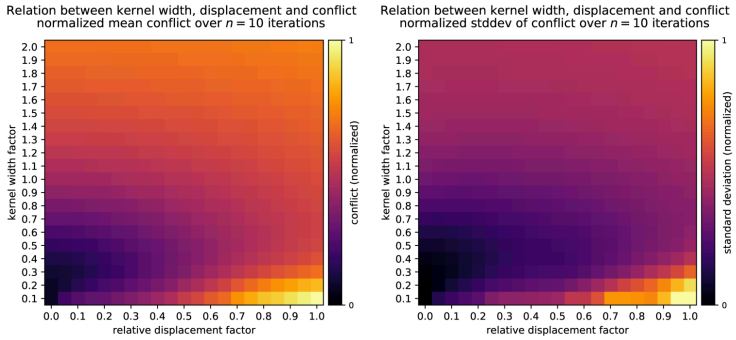
Parameters We initialize the scenario with the parameters given in Table 7.2.

Results The mean and the standard deviation of the conflict, calculated by Δ_{NW} and Δ_{RBFN} for `agent_stepsize` = 1.5 are visualized in Figs. 7.11a and 7.11b respectively. The results for other step sizes are given in Figs. D.1 and D.2.

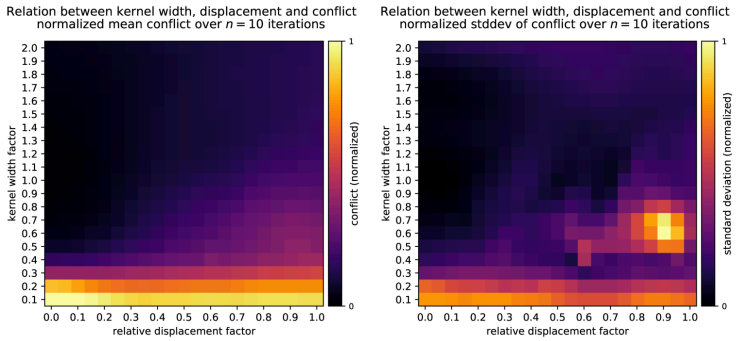
We identify a relation between the kernel widths of Δ_{NW}, Δ_{RBFN}, `agent_stepsize` and the intrinsic conflict. The aim is to find values for the kernel widths σ_{NW} and σ_{RBFN} that cause the output of both functions to be close to 0.0 in cases with no displacement and to monotonically increase with increasing 'disorder'. The relation between σ_{NW} and σ_{RBFN} is given as the parameter `conflict_sigma_factor`, denoted 'kernel width factor' in the figures.

For Δ_{NW} we see that for a small `DSF` of 0.1 to 0.2 the increasing output of Δ_{NW} corresponds to the increasing intrinsic conflict in the data. However, the standard deviation increases as well, indicating that the output of Δ_{NW} becomes less robust as the intrinsic conflict increases. For larger values of `conflict_sigma_factor`, 0.5 and above, the return value of Δ_{NW} however is almost constant, meaning that Δ_{NW} fails to capture the intrinsic conflict in the data. Considering the feedback across all values for `agent_stepsize` we identified `conflict_sigma_factor` = 0.3 as a compromise value for Δ_{NW}.

For Δ_{RBFN} and small `conflict_sigma_factor` of 0.1 to 0.4, the return value of is almost constant, i.e., it fails to capture the intrinsic conflict in the data. For larger values of `conflict_sigma_factor`, 0.5 to 0.8, the increasing output of Δ_{RBFN} corresponds to the increasing intrinsic conflict in the data. However, the standard deviation shows a peak for displacements around 0.8 to 1.0. A displacement factor of 1.0 corresponds to a displacement of `agent_stepsize`. As this is the distance between individual memory nodes in Ω_t, the output of Δ_{RBFN} should be robust for such a critical distance. For larger values of



(a) Mean and standard deviation of the Δ_{NW} conflict function for a step size of 1.5.



(b) Mean and standard deviation of the Δ_{RBFN} conflict function for a step size of 1.5.

Figure 7.11: Results from Scenario I, conflict function evaluation. Mean and standard deviation for the conflict functions Δ_{NW} and Δ_{RBFN} for a step size of 1.5. Normalization performed over all results in one figure to allow for comparison of results.

`conflict_sigma_factor`, 1.2 and above, the return value of Δ_{RBFN} however is almost constant, meaning that it in these cases fails to capture the intrinsic conflict in the data. Considering the feedback across all values for `agent_stepsize`we identified `conflict_sigma_factor` = 1.0 as a compromise value for Δ_{RBFN}.

7.4.1.1 Summary of Scenario I

We have shown that both Δ_{NW} and Δ_{RBFN} can detect the intrinsic conflict in a given set of data points. In both cases the return value of these functions correspond to the degree of disorder introduced into these sets.

We identified a suitable relation between the step size of the agent and the kernel widths of the conflict functions. In the next scenarios we will utilize these values, given as $\texttt{conflict\_sigma\_factor}_{NW} = 0.3$ and $\texttt{conflict\_sigma\_factor}_{RBFN} = 1.0$ respectively.

num_iterations	10
dim_controls	3 and 2
noise_actuator	0.0
enable_sensors	true
noise_sensors	0.0
agent_stepsize	0.5, 1.0 and 1.5
exploration_pattern	$C[S]$, $C[R]$, $C[SA]$ and $C[RA]$
conflict_function	Δ_{NW} and Δ_{RBFN}
conflict_sigma_factor	0.3 for Δ_{NW}, 1.0 for Δ_{RBFN} (Scenario I ,Section 7.4.1)
conflict_subsets	(1, 1.0)
bootstrapping_type	classify
pso_num_particles	–
pso_parameterization	–
hypothesis_group	$SE(2)$
hypothesis_set	$\mathrm{H}_1^{\oplus}\mathrm{SE}(2)$, $\mathrm{H}_2^{\oplus}\mathrm{SE}(2)$, $\mathrm{H}_1^{\ominus}\mathrm{SE}(2)$ and $\mathrm{H}_2^{\ominus}\mathrm{SE}(2)$

Table 7.3: Parameters for Scenario II, pattern and step size evaluation.

7.4.2 Scenario II : Holonomic and Nonholonomic Pattern Evaluation on SE(2)

In Scenario I (Section 7.4.1), we identified a suitable value for the parameter that relates the width of the kernels utilized in Δ_{NW} and Δ_{RBFN} to the agent step size. The next step is to evaluate the effect of both the agent step size and the type of exploration pattern on the performance of AgentLoop::Classify (see Algorithms 9 to 11). To that end we perform multiple experiments with varying combinations of step sizes and exploration patterns. With respect to the exploration patterns we will use the four patterns proposed in Section 7.2.4. The aim is to find a combination that maximizes the intrinsic conflict in the sensorimotor map and thus allows for a robust classification. We perform these experiments for the holonomic case using the hypothesis sets $\mathrm{H}_1^{\oplus}\mathrm{SE}(2)$ and $\mathrm{H}_2^{\oplus}\mathrm{SE}(2)$ and for the nonholonomic case using the hypothesis sets $\mathrm{H}_1^{\ominus}\mathrm{SE}(2)$ and $\mathrm{H}_2^{\ominus}\mathrm{SE}(2)$ respectively.

Parameters We initialize the scenario with the parameters given in Table 7.3.

Results The results for Δ_{NW} are depicted in Table 7.4 with additional figures and tables given in Appendices D.2.1 and D.3.1 In the holonomic case the classification accuracy is 1.0 in almost all cases as depicted in Tables 7.4a and 7.4b . The only exception is the combination of a step size of 0.5 in conjunction with $C[S]$ and $\mathrm{H}_2^{\oplus}\mathrm{SE}(2)$ as seen in Table 7.4b. In the nonholonomic case the classification accuracy varies with minimal values of 0.63 in Table 7.4c and 0.60 in Table 7.4d respectively. Moreover we see that only for step sizes 1.0 and 1.5 in combination with patterns $C[S]$ and $C[SA]$, AgentLoop::Classify succeeds in detecting the true hypothesis in all cases.

To distinguish between the results we consider the intrinsic conflict as proposed in Section 7.3.2. A general observation is that increasing the step size decreases the mean intrinsic conflict in all cases. In addition we see that the mean intrinsic conflict associated to simple patterns $C[S]$ and $C[SA]$ is lower than the mean intrinsic conflict associated to returning patterns $C[R]$ and $C[RA]$ with the same step size. Thus increasing the step size and utilizing simple instead of returning patterns makes the result of AgentLoop::Classify more robust. Looking at

Stepsize	Pattern	$\Delta[\Xi]/\sum I$		Accuracy
-	-	μ	σ	-
0.5	$C[S]$	0.502	0.283	1.00
0.5	$C[SA]$	0.511	0.284	1.00
0.5	$C[R]$	0.508	0.288	1.00
0.5	$C[RA]$	0.563	0.287	1.00
1.0	$C[S]$	0.363	0.256	1.00
1.0	$C[SA]$	0.337	0.254	1.00
1.0	$C[R]$	0.387	0.266	1.00
1.0	$C[RA]$	0.428	0.254	1.00
1.5	$C[S]$	0.271	0.242	1.00
1.5	$C[SA]$	0.258	0.243	1.00
1.5	$C[R]$	0.316	0.252	1.00
1.5	$C[RA]$	0.351	0.249	1.00

(a) Evaluation on $H_1^{\oplus}SE(2)$.

Stepsize	Pattern	$\Delta[\Xi]/\sum I$		Accuracy
-	-	μ	σ	-
0.5	$C[S]$	0.472	0.373	0.98
0.5	$C[SA]$	0.508	0.367	1.00
0.5	$C[R]$	0.530	0.374	1.00
0.5	$C[RA]$	0.588	0.369	1.00
1.0	$C[S]$	0.368	0.356	1.00
1.0	$C[SA]$	0.372	0.359	1.00
1.0	$C[R]$	0.472	0.355	1.00
1.0	$C[RA]$	0.482	0.351	1.00
1.5	$C[S]$	0.311	0.366	1.00
1.5	$C[SA]$	0.316	0.363	1.00
1.5	$C[R]$	0.338	0.365	1.00
1.5	$C[RA]$	0.397	0.355	1.00

(b) Evaluation on $H_2^{\oplus}SE(2)$.

Stepsize	Pattern	$\Delta[\Xi]/\sum I$		Accuracy
-	-	μ	σ	-
0.5	$C[S]$	0.583	0.363	1.00
0.5	$C[SA]$	0.563	0.363	0.96
0.5	$C[R]$	0.625	0.399	0.63
0.5	$C[RA]$	0.616	0.410	0.64
1.0	$C[S]$	0.468	0.340	1.00
1.0	$C[SA]$	0.459	0.348	1.00
1.0	$C[R]$	0.520	0.395	0.96
1.0	$C[RA]$	0.595	0.372	0.74
1.5	$C[S]$	0.386	0.339	1.00
1.5	$C[SA]$	0.384	0.345	1.00
1.5	$C[R]$	0.478	0.377	0.99
1.5	$C[RA]$	0.531	0.381	0.81

(c) Evaluation on $H_1^{\ominus}SE(2)$.

Stepsize	Pattern	$\Delta[\Xi]/\sum I$		Accuracy
-	-	μ	σ	-
0.5	$C[S]$	0.593	0.342	0.92
0.5	$C[SA]$	0.599	0.338	0.88
0.5	$C[R]$	0.706	0.357	0.62
0.5	$C[RA]$	0.682	0.345	0.60
1.0	$C[S]$	0.405	0.313	1.00
1.0	$C[SA]$	0.408	0.318	1.00
1.0	$C[R]$	0.550	0.344	1.00
1.0	$C[RA]$	0.603	0.329	0.94
1.5	$C[S]$	0.328	0.306	1.00
1.5	$C[SA]$	0.332	0.307	1.00
1.5	$C[R]$	0.460	0.326	1.00
1.5	$C[RA]$	0.486	0.311	0.93

(d) Evaluation on $H_2^{\ominus}SE(2)$.

Table 7.4: Results for Scenario II, pattern and step size evaluation utilizing Δ_{NW}. Evaluation on hypothesis sets $H_1^{\oplus}SE(2)$, $H_2^{\oplus}SE(2)$, $H_1^{\ominus}SE(2)$ and $H_2^{\oplus}SE(2)$ with various patterns and step sizes. Intrinsic conflict (μ and σ) and classification accuracy.

the intrinsic conflict associated to the different step sizes 1.0 and 1.5 and the patterns $C[S]$ and $C[SA]$ in Tables 7.4c and 7.4d, we see that for the larger step size 1.5 in all cases both the mean and standard deviation is smaller. To distinguish between $C[S]$ and $C[SA]$ we consider the mean and standard deviation of the intrinsic conflict for all scenarios from Table 7.4 that involve a step size of 1.5 and these two patterns. However, in this regard the differences between $C[S]$ and $C[SA]$ are only marginal. To come to a conclusion, we take the number of samples and obstructions stored in each neighborhood into account, as given in Tables D.2 and D.6. These two properties are denoted $\mathcal{S}/\Lambda[\Xi]$ and $\mathcal{B}/\Lambda[\Xi]$ respectively. Increasing the number of data points in a sensorimotor neighborhood, increases the potential for intrinsic conflict as it directly increases the number of elements the conflict function Δ_{NW} works with. In this regard $C[SA]$ has a slight advantage over $C[S]$.

The results for Δ_{RBFN} are depicted in Table 7.5 with additional figures and tables given in Appendices D.2.2 and D.3.2. In both the holonomic and the nonholonomic case the classification accuracy varies greatly with minimal values of 0.44 in Tables 7.5a and 7.5d and 0.76 in Table 7.5c. Only in Table 7.5b the classification accuracy is at or above 0.90 in all cases. Com-

Stepsize	Pattern	$\Delta[\Xi]/\sum I$		Accuracy
-	-	μ	σ	-
0.5	$C[S]$	0.535	0.284	0.95
0.5	$C[SA]$	0.541	0.289	0.95
0.5	$C[R]$	0.512	0.295	1.00
0.5	$C[RA]$	0.568	0.293	1.00
1.0	$C[S]$	0.621	0.320	0.61
1.0	$C[SA]$	0.627	0.314	0.64
1.0	$C[R]$	0.569	0.328	0.86
1.0	$C[RA]$	0.588	0.327	0.81
1.5	$C[S]$	0.600	0.344	0.45
1.5	$C[SA]$	0.615	0.333	0.44
1.5	$C[R]$	0.573	0.347	0.60
1.5	$C[RA]$	0.588	0.330	0.62

(a) Evaluation on $\mathrm{H}_1^{\oplus}\mathrm{SE}(2)$.

Stepsize	Pattern	$\Delta[\Xi]/\sum I$		Accuracy
-	-	μ	σ	-
0.5	$C[S]$	0.483	0.380	0.94
0.5	$C[SA]$	0.535	0.375	0.92
0.5	$C[R]$	0.517	0.376	1.00
0.5	$C[RA]$	0.559	0.378	0.90
1.0	$C[S]$	0.375	0.355	1.00
1.0	$C[SA]$	0.383	0.359	1.00
1.0	$C[R]$	0.442	0.356	1.00
1.0	$C[RA]$	0.435	0.363	1.00
1.5	$C[S]$	0.324	0.365	1.00
1.5	$C[SA]$	0.325	0.361	1.00
1.5	$C[R]$	0.343	0.362	1.00
1.5	$C[RA]$	0.356	0.360	1.00

(b) Evaluation on $\mathrm{H}_2^{\oplus}\mathrm{SE}(2)$.

Stepsize	Pattern	$\Delta[\Xi]/\sum I$		Accuracy
-	-	μ	σ	-
0.5	$C[S]$	0.588	0.358	0.89
0.5	$C[SA]$	0.598	0.343	0.88
0.5	$C[R]$	0.620	0.362	0.83
0.5	$C[RA]$	0.664	0.345	0.76
1.0	$C[S]$	0.706	0.324	0.81
1.0	$C[SA]$	0.734	0.331	0.79
1.0	$C[R]$	0.654	0.315	0.91
1.0	$C[RA]$	0.703	0.327	0.84
1.5	$C[S]$	0.739	0.332	0.70
1.5	$C[SA]$	0.741	0.337	0.68
1.5	$C[R]$	0.665	0.323	0.86
1.5	$C[RA]$	0.663	0.324	0.77

(c) Evaluation on $\mathrm{H}_1^{\ominus}\mathrm{SE}(2)$.

Stepsize	Pattern	$\Delta[\Xi]/\sum I$		Accuracy
-	-	μ	σ	-
0.5	$C[S]$	0.619	0.334	0.71
0.5	$C[SA]$	0.576	0.344	0.71
0.5	$C[R]$	0.698	0.338	0.53
0.5	$C[RA]$	0.638	0.344	0.44
1.0	$C[S]$	0.418	0.316	0.99
1.0	$C[SA]$	0.408	0.316	0.98
1.0	$C[R]$	0.479	0.332	0.98
1.0	$C[RA]$	0.480	0.332	0.90
1.5	$C[S]$	0.333	0.307	1.00
1.5	$C[SA]$	0.332	0.304	1.00
1.5	$C[R]$	0.344	0.326	1.00
1.5	$C[RA]$	0.373	0.324	0.99

(d) Evaluation on $\mathrm{H}_2^{\ominus}\mathrm{SE}(2)$.

Table 7.5: Results for Scenario II, pattern and step size evaluation utilizing Δ_{RBFN}. Evaluation on hypothesis sets $\mathrm{H}_1^{\oplus}\mathrm{SE}(2)$, $\mathrm{H}_2^{\oplus}\mathrm{SE}(2)$, $\mathrm{H}_1^{\ominus}\mathrm{SE}(2)$ and $\mathrm{H}_2^{\oplus}\mathrm{SE}(2)$ with various patterns and step sizes. Intrinsic conflict (μ and σ) and classification accuracy.

paring the results from Δ_{RBFN} to the results from Δ_{NW} we see that Δ_{RBFN} performs considerably worse.

Unlike for Δ_{NW}, the results depicted in Table 7.5 are conflicting and thus make identifying a suitable combination of step size and exploration pattern difficult. Table 7.5a suggests a small step size of 0.5 in combination with either $C[R]$ or $C[RA]$. These combinations, however, are those with the lowest classification accuracy in Table 7.5d and a low classification accuracy in Table 7.5c. While Table 7.5d suggests a step size of 1.5, combined with $C[S]$, $C[R]$ or $C[RA]$ the classification accuracy associated to these combinations given in Table 7.5c is low.

An interesting effect can be seen in the holonomic case utilizing the hypothesis set $\mathrm{H}_2^{\ominus}\mathrm{SE}(2)$. As depicted in Fig. 7.12 the output of Δ_{NW} for various combinations of predicted and true hypothesis is the same. Consider the column labeled with H_1 that corresponds to the results of the 10 iterations in which the agent initialized with the hypothesis H_1 has been evaluated. The rows H_1, H_4 and H_7 indicate that AgentLoop::Classify assessed the same intrinsic conflict to all these three hypotheses. The same effect can be observed in columns H_4 and H_7. This is due to the fact that for these hypotheses, and a given control $[\, c_1, c_2 \,]^T \in C_{\odot}$, both elements c_1 and c_1

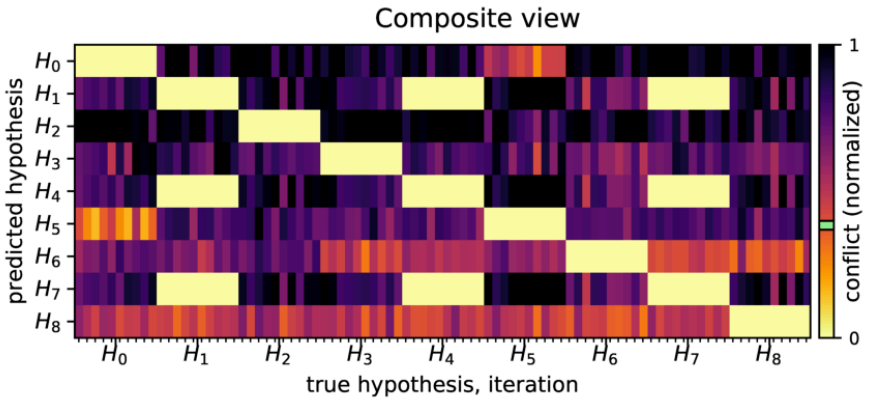


Figure 7.12: Example composite matrix from Scenario II, pattern and step size evaluation. Visualization of results from hypothesis set $\mathrm{H}_1^{\ominus}\mathrm{SE}(2)$, pattern $C[SA]$, and conflict function Δ_{NW} with a step size of 1.5.

are mapped to the first two generators $\hat{\mathfrak{g}}_1$ and $\hat{\mathfrak{g}}_2$ of $SE(2)$ (see Appendices A.2.1 and C.4). The hypotheses H_1, H_4 and H_7 only differ with respect to the weight vectors used in their creation (see Section 7.2.6). For H_1 the weight vector is given as $[0.5, 0.5]$, for H_4 it is $[1.0, 1.0]$ and for for H_7 it is $[1.5, 0.5]$. The generators $\hat{\mathfrak{g}}_1$ and $\hat{\mathfrak{g}}_2$ correspond to translation in x- and y-axis, respectively. As translation is commutative, scaling individual controls has no impact on the structure of the sensorimotor map. This means that for these three hypotheses, for arbitrary controls $c_i, c_{i+1}, \cdots, c_j$ from $C_{\odot} \subseteq \mathbb{R}^2$ it holds that

$$\tau_{\blacktriangleleft}(\tau_{\blacktriangleright}(c_i) \bullet \tau_{\blacktriangleright}(c_{i+1}) \bullet \cdots \bullet \tau_{\blacktriangleright}(c_j)) = c_i + c_{i+1} + \cdots + c_j = c_{ij} \quad \text{with} \quad i < j \tag{7.5}$$

where c_{ij} is from $C_* \equiv \mathbb{R}^3$ according to the definition of $\tau_{\blacktriangleright}$ and $\tau_{\blacktriangleleft}$ (see Eqs. (4.17) and (4.38) respectively). Thus the spatial relations between data points are identical, the sensorimotor neighborhoods are identical and naturally the output of the conflict function is identical as well.

However we have to discuss to which extent the scenario still qualifies as a nonholonomic system. In this case the configuration space of the agent is reduced to $\mathbb{R}^2$. With respect to $\mathbb{R}^2$ the agent is holonomic, as it can directly control it's two degrees of freedom. However with respect to $SE(2)$ it is nonholonomic, as the agent can not change its orientation, i.e. is it restricted to a subset of configurations from $SE(2)$.

7.4.2.1 Summary of Scenario II

Utilizing Δ_{NW}, AgentLoop::Classify can detect the true hypothesis in both the holonomic and the nonholonomic case and with respect to both sets of hypotheses. Increasing the step size and utilizing simple instead of returning patterns makes the result of AgentLoop::Classify more robust. For certain combinations of step size and exploration pattern the classification accuracy is 1.0. Taking into accounts the classification accuracy, the intrinsic conflict as well as the properties of the neighborhoods of the sensorimotor map, a combination of a step size of 1.5 in conjunction with the pattern $C[SA]$ has been selected to be utilized in the next scenarios.

num_iterations	10
dim_controls	6 and 2
noise_actuator	0.0
enable_sensors	true
noise_sensors	0.0
agent_stepsize	0.5, 1.0 and 1.5
exploration_pattern	$C[S]$, $C[R]$, $C[SA]$, and $C[RA]$
conflict_function	Δ_{NW} and Δ_{RBFN}
conflict_sigma_factor	0.3 for Δ_{NW}, 1.0 for Δ_{RBFN} (Scenario I ,Section 7.4.1)
conflict_subsets	(1, 1.0)
bootstrapping_type	classify
pso_num_particles	–
pso_parameterization	–
hypothesis_group	$SE(3)$
hypothesis_set	$\text{H}^{\oplus}\text{SE}(3)$ and $\text{H}^{\ominus}\text{SE}(3)$

Table 7.6: Parameters for Scenario III. pattern and step size evaluation.

The performance of AgentLoop::Classify using Δ_{RBFN} however is considerably worse when compared to Δ_{NW}. However, the main issue is that the results are not consistent. Results from the holonomic and nonholonomic scenarios and different hypothesis sets suggests different step sizes as well as different exploration patterns. This does not allow us to select a combination of step size and exploration pattern that performs well in all scenarios. Due to these results and the overall poor performance of Δ_{RBFN} we decided to not utilize it in the next $SE(2)$ scenarios.

7.4.3 Scenario III : Holonomic and Nonholonomic Pattern Evaluation on SE(3)

In this scenario we evaluate the effect both the agent step size and the type of exploration pattern has on the performance of AgentLoop::Classify (see Algorithms 9 to 11). We utilize a similar setup as in Scenario II (see Section 7.4.2) but with the difference that we use hypothesis sets based on $SE(3)$ instead of $SE(2)$. We perform these experiments for the holonomic case using the hypothesis set $\text{H}^{\oplus}\text{SE}(3)$ and for the nonholonomic case using the hypothesis set $\text{H}^{\ominus}\text{SE}(3)$ respectively.

Parameters We initialize the scenario with the parameters given in Table 7.6.

Results The results for Δ_{NW} are depicted in Tables 7.7a and 7.7b with additional figures and tables given in Appendices D.4.1 and D.5.1 In both the holonomic and nonholonomic case, the classification accuracy varies with minimal values of 0.43 in Table 7.7a, and 0.47 in Table 7.7b respectively. In the nonholonomic case, as depicted in in Table 7.7b, for step sizes 1.0 and 1.5 in combination with patterns $C[S]$ and $C[SA]$, AgentLoop::Classify succeeds in detecting the true hypothesis in all cases. However only for step sizes 1.5 in combination with patterns $C[S]$ and $C[SA]$, AgentLoop::Classify succeeds in detecting the true hypothesis in both holonomic and nonholonomic scenarios. Considering the mean intrinsic conflict to distinguish between $C[S]$ and $C[SA]$ we see that $C[S]$ performs slightly better than $C[SA]$. A general observation is that increasing the step size decreases the mean intrinsic conflict in most cases where like in Scenario II the mean intrinsic conflict associated to simple patterns $C[S]$ and $C[SA]$ is lower

Stepsize	Pattern	$\Delta[\Xi]/\sum I$		Accuracy
-	-	μ	σ	-
0.5	$C[S]$	0.568	0.287	0.56
0.5	$C[SA]$	0.567	0.287	0.44
0.5	$C[R]$	0.667	0.294	0.65
0.5	$C[RA]$	0.664	0.291	0.43
1.0	$C[S]$	0.431	0.258	0.95
1.0	$C[SA]$	0.463	0.268	0.93
1.0	$C[R]$	0.479	0.271	0.95
1.0	$C[RA]$	0.519	0.276	0.86
1.5	$C[S]$	0.318	0.241	1.00
1.5	$C[SA]$	0.349	0.245	1.00
1.5	$C[R]$	0.435	0.262	0.99
1.5	$C[RA]$	0.673	0.253	0.97

(a) Evaluation on $H^{\oplus}SE(3)$ using Δ_{NW}.

Stepsize	Pattern	$\Delta[\Xi]/\sum I$		Accuracy
-	-	μ	σ	-
0.5	$C[S]$	0.668	0.353	0.95
0.5	$C[SA]$	0.671	0.354	0.93
0.5	$C[R]$	0.748	0.332	0.47
0.5	$C[RA]$	0.766	0.343	0.64
1.0	$C[S]$	0.536	0.350	1.00
1.0	$C[SA]$	0.553	0.345	1.00
1.0	$C[R]$	0.650	0.366	0.68
1.0	$C[RA]$	0.666	0.355	0.74
1.5	$C[S]$	0.445	0.345	1.00
1.5	$C[SA]$	0.498	0.331	1.00
1.5	$C[R]$	0.609	0.378	0.70
1.5	$C[RA]$	0.622	0.355	0.72

(b) Evaluation on $H^{\ominus}SE(3)$ using Δ_{NW}.

Stepsize	Pattern	$\Delta[\Xi]/\sum I$		Accuracy
-	-	μ	σ	-
0.5	$C[S]$	0.527	0.287	0.21
0.5	$C[SA]$	0.514	0.289	0.12
0.5	$C[R]$	0.496	0.291	0.33
0.5	$C[RA]$	0.545	0.287	0.23
1.0	$C[S]$	0.481	0.285	0.72
1.0	$C[SA]$	0.475	0.284	0.59
1.0	$C[R]$	0.381	0.255	1.00
1.0	$C[RA]$	0.433	0.265	0.93
1.5	$C[S]$	0.378	0.253	0.99
1.5	$C[SA]$	0.379	0.249	0.99
1.5	$C[R]$	0.311	0.239	1.00
1.5	$C[RA]$	0.386	0.238	1.00

(c) Evaluation on $H^{\oplus}SE(3)$ using Δ_{RBFN}.

Stepsize	Pattern	$\Delta[\Xi]/\sum I$		Accuracy
-	-	μ	σ	-
0.5	$C[S]$	0.625	0.364	0.74
0.5	$C[SA]$	0.626	0.364	0.68
0.5	$C[R]$	0.650	0.347	0.68
0.5	$C[RA]$	0.654	0.338	0.62
1.0	$C[S]$	0.504	0.358	0.99
1.0	$C[SA]$	0.526	0.360	0.96
1.0	$C[R]$	0.493	0.363	0.99
1.0	$C[RA]$	0.542	0.356	0.95
1.5	$C[S]$	0.623	0.320	1.00
1.5	$C[SA]$	0.571	0.317	1.00
1.5	$C[R]$	0.590	0.314	1.00
1.5	$C[RA]$	0.470	0.329	1.00

(d) Evaluation on $H^{\ominus}SE(3)$ using Δ_{RBFN}.

Table 7.7: Results for Scenario III, pattern and step size evaluation. Evaluation on hypothesis sets $H^{\oplus}SE(3)$ and $H^{\ominus}SE(3)$ utilizing Δ_{NW} and Δ_{RBFN} with various patterns and step sizes. Intrinsic conflict (μ and σ) and classification accuracy.

than the mean intrinsic conflict associated to returning patterns $C[R]$ and $C[RA]$ with the same step size.

The results for Δ_{RBFN} are depicted in Tables 7.7c and 7.7d with additional figures and tables given in Appendices D.4.2 and D.5.2. In both the holonomic and nonholonomic case, the classification accuracy varies with minimal values of 0.12 in Table 7.7c and 0.47 in Table 7.7d respectively. However, for a step size of 1.5 and any exploration pattern, AgentLoop::Classify succeeds in detecting the true hypothesis in almost all cases, with a classification accuracy of 0.99 in Table 7.7c for $C[S]$ and $C[SA]$ as an exception. Like observed in Scenario II, also in this scenario when utilizing Δ_{NW}, increasing the step size decreases the mean intrinsic conflict. However only for step sizes 1.5 in combination with patterns $C[R]$ and $C[RA]$, AgentLoop::Classify succeeds in detecting the true hypothesis in both holonomic and nonholonomic scenarios. However the results in Table 7.7c and Table 7.7d do not allow for a definitive assessment whether $C[R]$ or $C[RA]$ is to be preferred. While based on the mean intrinsic conflict Table 7.7c suggests $C[R]$, Table 7.7d suggests $C[RA]$.

7.4.3.1 Summary of Scenario III

Utilizing Δ_{NW}, AgentLoop::Classify can detect the true hypothesis in both the holonomic and the nonholonomic case and with respect to both sets of hypotheses. Similar to the observations in Scenario II , increasing the step size and utilizing simple instead of returning patterns makes the result of AgentLoop::Classify more robust in most cases. For certain combinations of step size and exploration pattern the classification accuracy is 1.0. Taking into accounts the classification accuracy and the intrinsic conflict, a combination of a step size of 1.5 in conjunction with the pattern $C[S]$ has been identified as performing best.

In this scenario AgentLoop::Classify can detect the true hypothesis utilizing Δ_{RBFN} where in the best cases classification accuracy was 1.0 as well. Thus for both the holonomic and the nonholonomic cases it behaves differently, i.e. better, than observed in Scenario II. The performance of AgentLoop::Classify was best when utilizing a step size of 1.5. With respect to the exploration patterns, $C[R]$ and $C[RA]$ performed similarly well.

7.4.4 Scenario IV : Holonomic and Nonholonomic Baseline

After having identified the kernel sizes for the individuals conflict functions in Scenario I and a good combination of agent step size and exploration pattern in Scenario II we now perform a set of evaluations that aim on providing a baseline for both non-noisy scenarios and scenarios where the actuator and the sensors are subject to noise.

To that end we calculate the intrinsic conflict of a given sensorimotor chain with respect to an existing ground truth map of the environment. This map consists of a single extended sensorimotor neighborhood $\Lambda^{GT}_{\circledcirc}[\xi_1]$ with no limit set to $\epsilon_{\circledcirc}$ where for any $c \in C_{\circledcirc}$ ground truth sample and occupancy information can be calculated. Conflict for a given sensorimotor chain is calculated based on the single sensorimotor projection set $\Lambda^{GT}_{\circledcirc}[\xi_1 \uplus \Omega_t]$ (see Eq. (5.97)) as

$$
\begin{aligned}
\Delta_{GT}[\Omega_t] = \frac{1}{N_i}\sum(\, s_i - \Pi(\tau_{\triangleright}(c_i))\,)^2 + \frac{1}{N_j}\sum(\beta^*_j - \varrho(\tau_{\triangleright}(c_j))^*\,)^2 \quad \text{where}\\
\forall(c_i, s_i) \in \Lambda^{GT}_{(\circledcirc,\mathcal{S})}[\xi_1 \uplus \Omega_t],\\
\forall(c_j, \beta_j) \in \Lambda^{GT}_{(\circledcirc,\mathcal{B})}[\xi_1 \uplus \Omega_t],\\
N_i = |\ \in \Lambda^{GT}_{(\circledcirc,\mathcal{S})}[\xi_1 \uplus \Omega_t]\,|,\\
N_j = |\ \in \Lambda^{GT}_{(\circledcirc,\mathcal{B})}[\xi_1 \uplus \Omega_t]\,|.
\end{aligned}
\tag{7.6}
$$

Here β^* denotes the real-valued equivalent for $\beta_\bullet$ and $\beta_\circ$ with $\beta^*_\bullet = 0.0$ and $\beta^*_\circ = 1.0$ respectively as introduced in Section 6.2.2.1. As this function is supposed to provide a ground truth reference, the sample function Π (see Eq. (4.52)) is assumed to be non-noisy. If the agent's actuator and sensors are non-noisy, the first term containing the difference between the measured sample s_i and the sample provided by the ground truth map $\Pi(\tau_{\triangleright}(c_i))$ will be zero in all cases. However, for the second term that calculates the difference between occupancy data, this is not always true. It holds that $\beta^*_j - \varrho(\tau_{\triangleright}(c_j)^*$ will be zero whenever $\beta^*_j = \beta^*_\circ$. However, for cases where $\beta^*_j = \beta^*_\bullet$ this might be not the case. The reason for that is that obstructions are encoded as the difference between scheduled and effective controls as given by Eqs. (4.58) to (4.60). A path between

num_iterations	5
dim_controls	3 and 2
noise_actuator	0.0
enable_sensors	true
noise_sensors	0.0
agent_stepsize	1.5 (Scenario II ,Section 7.4.2)
exploration_pattern	$C[SA]$ (Scenario II ,Section 7.4.2)
conflict_function	$\Delta_{\mathrm{GT}}[\Omega_t]$
conflict_sigma_factor	–
conflict_subsets	(1, 1.0)
bootstrapping_type	classify
pso_num_particles	–
pso_parameterization	–
hypothesis_group	$SE(2)$
hypothesis_set	$\mathrm{H}_1^{\oplus}\mathrm{SE}(2)$, $\mathrm{H}_2^{\oplus}\mathrm{SE}(2)$, $\mathrm{H}_1^{\ominus}\mathrm{SE}(2)$ and $\mathrm{H}_2^{\ominus}\mathrm{SE}(2)$

Table 7.8: Parameters for Scenario IV, non-noisy actuator and sensor evaluation.

q_i, q_j on $\mathcal{Q}^n$ that is described by $c_i^{\ominus}$ might be obstructed in which case $c_i^{\ominus}$ differs from $c_i^{\ominus}$. While this means that for some configuration q' on that path it holds that $\varrho(q') = \beta_\bullet$, we can't make any assumption regarding q_j. Thus $\varrho(q_j) = \beta_\bullet$ might or might not be true. Correspondingly $\beta_j^* - \varrho(\tau_\triangleright(c_j)^*$ will be nonzero in situations where the agent encountered an obstruction on its path, while the destination itself has been unobstructed. This difference between 'obstructed paths' versus 'obstructed configurations' will introduce a minimal conflict into Δ_{GT} for these cases.

The outcome of these calculations will serve as a baseline for Scenario V and Scenario VI. We will calculate $\Delta_{GT}[\Omega_t]$ for four distinct cases. In the first case given in Section 7.4.4.1 actuator and sensors are parameterized to be noise-free. In the second and third case given in Sections 7.4.4.2 and 7.4.4.3 either the sensors or the actuator are parameterized to be noisy. In the fourth case given in Section 7.4.4.4 both the actuator and the sensors are noisy. We perform these experiments for the holonomic case using the hypothesis sets $\mathrm{H}_1^{\oplus}\mathrm{SE}(2)$ and $\mathrm{H}_2^{\oplus}\mathrm{SE}(2)$ and for the nonholonomic case using the hypothesis sets $\mathrm{H}_1^{\ominus}\mathrm{SE}(2)$ and $\mathrm{H}_2^{\ominus}\mathrm{SE}(2)$ respectively.

7.4.4.1 Non-Noisy Actuator and Sensor Evaluation

Parameters We initialize the scenario with the parameters given in Table 7.8.

Results The results are depicted in Table 7.9 with additional figures and tables given in Appendices D.6.1 and D.7.1. The classification accuracy is 1.0 in all cases which is expected as the sensorimotor chain is compared to the ground truth environment information.

7.4.4.2 Sensor Noise Evaluation

Parameters We initialize the scenario with the parameters given in Table 7.10.

Results The results are depicted in Table 7.11. The classification accuracy stays at 1.0 for all cases. Though the mean intrinsic conflict increases slightly at higher noise levels, sensor

Noise	$\Delta[\Xi]/\sum I$		Accuracy
σ_c, σ_p	μ	σ	-
0.000	0.145	0.245	1.00

(a) Evaluation on $H_1^{\oplus}SE(2)$.

Noise	$\Delta[\Xi]/\sum I$		Accuracy
σ_c, σ_p	μ	σ	-
0.000	0.255	0.377	1.00

(b) Evaluation on $H_2^{\oplus}SE(2)$.

Noise	$\Delta[\Xi]/\sum I$		Accuracy
σ_c, σ_p	μ	σ	-
0.000	0.192	0.294	1.00

(c) Evaluation on $H_1^{\ominus}SE(2)$.

Noise	$\Delta[\Xi]/\sum I$		Accuracy
σ_c, σ_p	μ	σ	-
0.000	0.188	0.296	1.00

(d) Evaluation on $H_2^{\ominus}SE(2)$.

Table 7.9: Results for Scenario IV, non-noisy actuator and sensor evaluation. Evaluation on hypothesis sets $H_1^{\oplus}SE(2)$, $H_2^{\oplus}SE(2)$, $H_1^{\ominus}SE(2)$, and $H_2^{\ominus}SE(2)$. Intrinsic conflict (μ and σ) and classification accuracy.

`num_iterations`	5
`dim_controls`	3 and 2
`noise_actuator`	0.0
`enable_sensors`	`true`
`noise_sensors`	0.001, 0.0025, 0.005, 0.01, 0.025, 0.05, 0.1, 0.25, 0.5, 0.75 and 1.0
`agent_stepsize`	1.5 (Scenario II ,Section 7.4.2)
`exploration_pattern`	$C[SA]$ (Scenario II ,Section 7.4.2)
`conflict_function`	$\Delta_{GT}[\Omega_t]$
`conflict_sigma_factor`	–
`conflict_subsets`	(1, 1.0)
`bootstrapping_type`	`classify`
`pso_num_particles`	–
`pso_parameterization`	–
`hypothesis_group`	$SE(2)$
`hypothesis_set`	$H_1^{\oplus}SE(2)$, $H_2^{\oplus}SE(2)$, $H_1^{\ominus}SE(2)$ and $H_2^{\ominus}SE(2)$

Table 7.10: Parameters for Scenario IV, sensor noise evaluation.

noise does not negatively affect classification accuracy in these cases. Considering the way that sensor noise is calculated (see Eqs. (4.49) and (4.50)) and the way that $\Delta_{GT}[\Omega_t]$ is calculated (see Eq. (7.6)) this is an expected effect. Noisy samples are created by adding a zero-mean Gaussian noise to the ground truth measurements. The conflict function calculates the mean squared error over all samples based on the ground truth measurements. Though this value inevitably increases when increasing σ_p, if considering enough samples, it will always be minimal for the true hypothesis.

7.4.4.3 Actuator Noise Evaluation

Parameters We initialize the scenario with the parameters given in Table 7.12.

Results The results are depicted in Table 7.13. with additional figures and tables given in Appendices D.6.3 and D.7.3 In both the holonomic and the nonholonomic case the classification accuracy remains relatively stable up until $\sigma_c = 0.01$. Increasing the noise further causes the classification accuracy to drop continuously until reaching chance level accuracy for the indi-

Noise	$\Delta[\Xi]/\sum I$		Accuracy
σ_p	μ	σ	-
0.001	0.145	0.245	1.00
0.003	0.145	0.245	1.00
0.005	0.145	0.245	1.00
0.010	0.145	0.245	1.00
0.025	0.145	0.245	1.00
0.050	0.145	0.245	1.00
0.100	0.145	0.244	1.00
0.250	0.145	0.244	1.00
0.500	0.147	0.244	1.00
0.750	0.151	0.243	1.00
1.000	0.155	0.242	1.00

(a) Evaluation on $\mathrm{H}_1^{\oplus}\mathrm{SE}(2)$.

Noise	$\Delta[\Xi]/\sum I$		Accuracy
σ_p	μ	σ	-
0.001	0.255	0.377	1.00
0.003	0.255	0.377	1.00
0.005	0.255	0.377	1.00
0.010	0.255	0.377	1.00
0.025	0.255	0.377	1.00
0.050	0.255	0.377	1.00
0.100	0.255	0.377	1.00
0.250	0.254	0.378	1.00
0.500	0.253	0.378	1.00
0.750	0.255	0.378	1.00
1.000	0.258	0.377	1.00

(b) Evaluation on $\mathrm{H}_2^{\oplus}\mathrm{SE}(2)$.

Noise	$\Delta[\Xi]/\sum I$		Accuracy
σ_p	μ	σ	-
0.001	0.192	0.294	1.00
0.003	0.192	0.294	1.00
0.005	0.192	0.294	1.00
0.010	0.192	0.294	1.00
0.025	0.192	0.294	1.00
0.050	0.192	0.294	1.00
0.100	0.192	0.294	1.00
0.250	0.192	0.294	1.00
0.500	0.194	0.294	1.00
0.750	0.197	0.293	1.00
1.000	0.201	0.292	1.00

(c) Evaluation on $\mathrm{H}_1^{\ominus}\mathrm{SE}(2)$.

Noise	$\Delta[\Xi]/\sum I$		Accuracy
σ_p	μ	σ	-
0.001	0.188	0.296	1.00
0.003	0.188	0.296	1.00
0.005	0.188	0.296	1.00
0.010	0.188	0.296	1.00
0.025	0.188	0.296	1.00
0.050	0.188	0.296	1.00
0.100	0.188	0.296	1.00
0.250	0.189	0.296	1.00
0.500	0.191	0.295	1.00
0.750	0.195	0.294	1.00
1.000	0.200	0.293	1.00

(d) Evaluation on $\mathrm{H}_2^{\ominus}\mathrm{SE}(2)$.

Table 7.11: Results for Scenario IV, sensor noise evaluation. Evaluation on hypothesis sets $\mathrm{H}_1^{\oplus}\mathrm{SE}(2)$, $\mathrm{H}_2^{\oplus}\mathrm{SE}(2)$, $\mathrm{H}_1^{\ominus}\mathrm{SE}(2)$ and $\mathrm{H}_2^{\ominus}\mathrm{SE}(2)$. Intrinsic conflict ($\mu$ and σ) and classification accuracy.

vidual hypothesis sets at $\sigma_c = 0.75$ given as $\frac{1}{14} = 0.07$ for $\mathrm{H}_1^{\oplus}\mathrm{SE}(2)$, $\frac{1}{5} = 0.25$ for $\mathrm{H}_1^{\oplus}\mathrm{SE}(2)$ and $\frac{1}{9} = 0.11$ for both $\mathrm{H}_1^{\ominus}\mathrm{SE}(2)$ and $\mathrm{H}_2^{\ominus}\mathrm{SE}(2)$.

7.4.4.4 Actuator and Sensor Noise Evaluation

Parameters We initialize the scenario with the parameters given in Table 7.14.

Results The results are depicted in Table 7.15 with additional figures and tables given in Appendices D.6.4 and D.7.4. In both the holonomic and the nonholonomic case the classification accuracy remains relatively stable up until $\sigma_c = 0.01$ with $\mathrm{H}_2^{\oplus}\mathrm{SE}(2)$ in Table 7.15b being the exception, starting to decline only after $\sigma_c = 0.025$. Increasing the noise further causes the classification accuracy to drop until reaching chance level accuracy for the individual hypothesis sets at around $\sigma_c = 0.75$ to $\sigma_c = 1.00$.

num_iterations	5
dim_controls	3 and 2
noise_actuator	0.001, 0.0025, 0.005, 0.01, 0.025, 0.05, 0.1, 0.25, 0.5, 0.75 and 1.0
enable_sensors	true
noise_sensors	0.0
agent_stepsize	1.5 (Scenario II ,Section 7.4.2)
exploration_pattern	$C[SA]$ (Scenario II ,Section 7.4.2)
conflict_function	$\Delta_{\mathrm{GT}}[\Omega_t]$
conflict_sigma_factor	–
conflict_subsets	(1, 1.0)
bootstrapping_type	classify
pso_num_particles	–
pso_parameterization	–
hypothesis_group	$SE(2)$
hypothesis_set	$\mathrm{H}_1^{\oplus}\mathrm{SE}(2)$, $\mathrm{H}_2^{\oplus}\mathrm{SE}(2)$, $\mathrm{H}_1^{\ominus}\mathrm{SE}(2)$ and $\mathrm{H}_2^{\ominus}\mathrm{SE}(2)$

Table 7.12: Parameters for Scenario IV, actuator noise evaluation.

Noise	$\Delta[\Xi]/\sum I$		Accuracy
σ_c	μ	σ	-
0.001	0.156	0.244	1.00
0.003	0.160	0.243	1.00
0.005	0.174	0.243	1.00
0.010	0.197	0.250	0.99
0.025	0.279	0.272	0.84
0.050	0.357	0.296	0.66
0.100	0.401	0.295	0.44
0.250	0.449	0.307	0.20
0.500	0.434	0.303	0.21
0.750	0.424	0.302	0.07
1.000	0.428	0.304	0.07

(a) Evaluation on $\mathrm{H}_1^{\oplus}\mathrm{SE}(2)$.

Noise	$\Delta[\Xi]/\sum I$		Accuracy
σ_c	μ	σ	-
0.001	0.266	0.374	1.00
0.003	0.263	0.377	1.00
0.005	0.262	0.377	1.00
0.010	0.276	0.373	1.00
0.025	0.340	0.359	1.00
0.050	0.413	0.379	0.76
0.100	0.446	0.369	0.60
0.250	0.481	0.382	0.16
0.500	0.461	0.377	0.32
0.750	0.482	0.385	0.24
1.000	0.484	0.379	0.24

(b) Evaluation on $\mathrm{H}_2^{\oplus}\mathrm{SE}(2)$.

Noise	$\Delta[\Xi]/\sum I$		Accuracy
σ_c	μ	σ	-
0.001	0.197	0.293	1.00
0.003	0.204	0.294	1.00
0.005	0.213	0.296	1.00
0.010	0.233	0.295	1.00
0.025	0.290	0.311	0.87
0.050	0.349	0.324	0.76
0.100	0.401	0.325	0.49
0.250	0.411	0.327	0.38
0.500	0.426	0.336	0.16
0.750	0.445	0.328	0.11
1.000	0.412	0.327	0.09

(c) Evaluation on $\mathrm{H}_1^{\ominus}\mathrm{SE}(2)$.

Noise	$\Delta[\Xi]/\sum I$		Accuracy
σ_c	μ	σ	-
0.001	0.194	0.295	1.00
0.003	0.203	0.294	1.00
0.005	0.211	0.294	1.00
0.010	0.233	0.295	1.00
0.025	0.312	0.315	0.93
0.050	0.366	0.313	0.78
0.100	0.405	0.329	0.47
0.250	0.443	0.328	0.29
0.500	0.458	0.322	0.16
0.750	0.467	0.332	0.11
1.000	0.454	0.328	0.11

(d) Evaluation on $\mathrm{H}_2^{\ominus}\mathrm{SE}(2)$.

Table 7.13: Results for Scenario IV, actuator noise evaluation. Evaluation on hypothesis sets $\mathrm{H}_1^{\oplus}\mathrm{SE}(2)$, $\mathrm{H}_2^{\oplus}\mathrm{SE}(2)$, $\mathrm{H}_1^{\ominus}\mathrm{SE}(2)$ and $\mathrm{H}_2^{\ominus}\mathrm{SE}(2)$. Intrinsic conflict ($\mu$ and σ) and classification accuracy.

num_iterations	5
dim_controls	3 and 2
noise_actuator	0.001, 0.0025, 0.005, 0.01, 0.025, 0.05, 0.1, 0.25, 0.5, 0.75 and 1.0
enable_sensors	true
noise_sensors	sensor noise (all sensors) matches actuator noise
agent_stepsize	1.5 (Scenario II ,Section 7.4.2)
exploration_pattern	$C[SA]$ (Scenario II ,Section 7.4.2)
conflict_function	$\Delta_{GT}[\Omega_t]$
conflict_sigma_factor	–
conflict_subsets	(1, 1.0)
bootstrapping_type	classify
pso_num_particles	–
pso_parameterization	–
hypothesis_group	$SE(2)$
hypothesis_set	$H_1^{\oplus}SE(2)$, $H_2^{\oplus}SE(2)$, $H_1^{\ominus}SE(2)$ and $H_2^{\ominus}SE(2)$

Table 7.14: Parameters for Scenario IV, actuator and sensor noise evaluation.

Noise	$\Delta[\Xi]/\sum I$		Accuracy
σ_c, σ_p	μ	σ	-
0.001	0.156	0.244	1.00
0.003	0.160	0.243	1.00
0.005	0.174	0.243	1.00
0.010	0.197	0.250	0.99
0.025	0.278	0.272	0.84
0.050	0.357	0.296	0.66
0.100	0.402	0.295	0.43
0.250	0.445	0.307	0.20
0.500	0.437	0.305	0.17
0.750	0.417	0.298	0.10
1.000	0.448	0.299	0.06

(a) Evaluation on $H_1^{\oplus}SE(2)$.

Noise	$\Delta[\Xi]/\sum I$		Accuracy
σ_c, σ_p	μ	σ	-
0.001	0.266	0.374	1.00
0.003	0.263	0.377	1.00
0.005	0.261	0.377	1.00
0.010	0.277	0.373	1.00
0.025	0.340	0.359	1.00
0.050	0.412	0.377	0.76
0.100	0.448	0.372	0.60
0.250	0.475	0.380	0.28
0.500	0.459	0.376	0.32
0.750	0.469	0.362	0.20
1.000	0.484	0.381	0.24

(b) Evaluation on $H_2^{\oplus}SE(2)$.

Noise	$\Delta[\Xi]/\sum I$		Accuracy
σ_c, σ_p	μ	σ	-
0.001	0.197	0.293	1.00
0.003	0.204	0.294	1.00
0.005	0.213	0.296	1.00
0.010	0.233	0.295	1.00
0.025	0.290	0.311	0.87
0.050	0.349	0.324	0.76
0.100	0.401	0.326	0.51
0.250	0.417	0.327	0.36
0.500	0.431	0.334	0.18
0.750	0.451	0.336	0.16
1.000	0.432	0.331	0.09

(c) Evaluation on $H_1^{\ominus}SE(2)$.

Noise	$\Delta[\Xi]/\sum I$		Accuracy
σ_c, σ_p	μ	σ	-
0.001	0.194	0.295	1.00
0.003	0.203	0.294	1.00
0.005	0.211	0.294	1.00
0.010	0.233	0.295	1.00
0.025	0.312	0.315	0.93
0.050	0.366	0.312	0.80
0.100	0.406	0.329	0.49
0.250	0.441	0.324	0.29
0.500	0.466	0.322	0.16
0.750	0.454	0.323	0.11
1.000	0.469	0.327	0.09

(d) Evaluation on $H_2^{\ominus}SE(2)$.

Table 7.15: Results for Scenario IV, actuator and sensor noise evaluation. Evaluation on hypothesis sets $H_1^{\oplus}SE(2)$, $H_2^{\oplus}SE(2)$, $H_1^{\ominus}SE(2)$ and $H_2^{\ominus}SE(2)$. Intrinsic conflict (μ and σ) and classification accuracy.

7.4.4.5 Summary of Scenario IV

In both the holonomic and the nonholonomic case, sensor noise did not negatively affect the classification accuracy. The effect of actuator noise, both individually and in combination with sensor noise, was similar for both holonomic and nonholonomic cases. Classification accuracy remained relatively stable up until $\sigma = 0.01$ to $\sigma = 0.025$. Increasing the noise further caused the classification accuracy to drop continuously until reaching chance level accuracy for the individual hypothesis sets at $\sigma = 0.75$ to $\sigma = 1.00$

7.4.5 Scenario V : Holonomic and Nonholonomic Classification

After having identified the kernel sizes for the individuals conflict functions in Scenario I and a good combination of agent step size and exploration pattern in Scenario II we now evaluate the performance of AgentLoop::Classify with respect to four questions:

- How does the classification accuracy change over time, i.e. how does the effect additional data points have on it, change over the course of an experiment?
- Is it possible to achieve a comparable classification accuracy while only utilizing a subset of data points from each neighborhood?
- To what extent is the classification accuracy affected by actuator and sensor noise? How does the performance of AgentLoop::Classify compare to the baseline calculated in Scenario IV?
- How much does every sensor contribute to the classification result, i.e. how does the classification accuracy change when enabling and disabling certain sensors?

We investigate these four questions in four different scenarios. All of them are performed using AgentLoop::Classify in conjunction with Δ_{NW}. We use the kernel parameters, the step size and the exploration patterns as identified in Scenario I and Scenario II respectively. We perform these experiments for the holonomic case using the hypothesis sets $H_1^{\oplus}SE(2)$ and $H_2^{\oplus}SE(2)$ and for the nonholonomic case using the hypothesis sets $H_1^{\ominus}SE(2)$ and $H_2^{\ominus}SE(2)$ respectively.

7.4.5.1 Intermediate Conflict Calculation

In the first scenario, we investigate the relation between classification accuracy and the number of data points over the curse of time. To that end and we evaluate the sensorimotor map intermediately every 50 steps, i.e. after every 50 cycles of the agent loop. This allows us to get an estimate on the whether adding additional data points increases the performance linearly over time or whether the effect additional data points have on classification accuracy and mean intrinsic conflict 'saturates' at some point.

Parameters We initialize the scenario with the parameters given in Table 7.16.

num_iterations	5
dim_controls	3 and 2
noise_actuator	0.0
enable_sensors	true
noise_sensors	0.0
agent_stepsize	1.5 (Scenario II ,Section 7.4.2)
exploration_pattern	$C[SA]$ (Scenario II ,Section 7.4.2)
conflict_function	Δ_{NW}
conflict_sigma_factor	0.3 (Scenario I ,Section 7.4.1)
conflict_subsets	(1, 1.0)
bootstrapping_type	classify
pso_num_particles	–
pso_parameterization	–
hypothesis_group	$SE(2)$
hypothesis_set	$\mathrm{H}_1^{\oplus}\mathrm{SE}(2)$, $\mathrm{H}_2^{\oplus}\mathrm{SE}(2)$, $\mathrm{H}_1^{\ominus}\mathrm{SE}(2)$ and $\mathrm{H}_2^{\ominus}\mathrm{SE}(2)$

Table 7.16: Parameters for Scenario V, intermediate conflict calculation.

Results The results are depicted in Table 7.17 with additional figures and tables given in Appendices D.8.1 and D.9.1. The classification accuracy at time step 50 is similarly low with 0.5 for all cases depicted in Table 7.17, with a classification accuracy of 0.71 in the nonholonomic case for $\mathrm{H}_1^{\oplus}\mathrm{SE}(2)$ depicted in Table 7.17c as an exception. A classification of 1.0 is reached after 250 time steps for $\mathrm{H}_1^{\oplus}\mathrm{SE}(2)$ and $\mathrm{H}_2^{\oplus}\mathrm{SE}(2)$ as depicted in Tables 7.17a and 7.17b respectively. For $\mathrm{H}_1^{\ominus}\mathrm{SE}(2)$ classification accuracy of 1.0 is reached at time step 300 as given in Table 7.17c where for $\mathrm{H}_2^{\ominus}\mathrm{SE}(2)$ this is not reached until time step 500 as shown in Table 7.17d. After that the mean intrinsic conflict decreases, indicating that the result of AgentLoop::Classify becomes more robust with additional data points added to the sensorimotor maps. The impact additional data points have on the mean intrinsic conflict diminishes slightly after time step 900 for $\mathrm{H}_1^{\oplus}\mathrm{SE}(2)$ and $\mathrm{H}_2^{\oplus}\mathrm{SE}(2)$ as depicted in Tables 7.17a and 7.17b respectively. For for $\mathrm{H}_1^{\ominus}\mathrm{SE}(2)$ and $\mathrm{H}_2^{\ominus}\mathrm{SE}(2)$ this happens after time step 800 as given in Tables 7.17c and 7.17d.

7.4.5.2 Subset Conflict Calculation

In the second scenario we will investigate the effect the number of utilized data points have on the performance of the conflict function and thus on the classification result. To that end we will select fractions of the data points we evaluate the conflict function on. As described in Section 7.2.5.1, the reduced number of data points we evaluate in Δ_{NW} will be compensated by executing the conflict function multiple times. Our aim is to evaluate whether a comparable classification accuracy can be maintained by simultaneously decreasing computation time given as T_{CPU} and T_{WALL}.

Parameters We initialize the scenario with the parameters given in Table 7.18.

Results The results are depicted in Tables 7.19 and 7.20 and Fig. 7.13 with additional figures and tables given in Appendices D.8.2 and D.9.2. For all cases, the classification accuracy remains at or slightly below 1.0 until $n = 8$, with the minimal classification accuracy of 0.9 for

Timestep	$\Delta[\Xi]/\sum I$		Accuracy
-	μ	σ	-
50	0.562	0.301	0.49
100	0.496	0.289	0.80
150	0.458	0.277	0.91
200	0.410	0.264	0.99
250	0.376	0.256	1.00
300	0.361	0.252	1.00
350	0.344	0.253	1.00
400	0.332	0.252	1.00
450	0.323	0.252	1.00
500	0.314	0.251	1.00
550	0.304	0.251	1.00
600	0.302	0.248	1.00
650	0.294	0.245	1.00
700	0.289	0.245	1.00
750	0.284	0.244	1.00
800	0.277	0.243	1.00
850	0.271	0.243	1.00
900	0.263	0.243	1.00
950	0.261	0.243	1.00
1000	0.259	0.242	1.00

(a) Evaluation on $H_1^{\oplus}SE(2)$.

Timestep	$\Delta[\Xi]/\sum I$		Accuracy
-	μ	σ	-
50	0.533	0.387	0.56
100	0.503	0.387	0.76
150	0.458	0.377	0.92
200	0.420	0.379	0.96
250	0.411	0.383	1.00
300	0.416	0.372	1.00
350	0.393	0.373	1.00
400	0.384	0.364	1.00
450	0.375	0.364	1.00
500	0.369	0.359	1.00
550	0.357	0.361	1.00
600	0.353	0.362	1.00
650	0.349	0.360	1.00
700	0.341	0.360	1.00
750	0.332	0.362	1.00
800	0.329	0.360	1.00
850	0.320	0.363	1.00
900	0.313	0.365	1.00
950	0.310	0.365	1.00
1000	0.311	0.364	1.00

(b) Evaluation on $H_2^{\oplus}SE(2)$.

Timestep	$\Delta[\Xi]/\sum I$		Accuracy
-	μ	σ	-
50	0.573	0.371	0.71
100	0.515	0.355	0.87
150	0.480	0.365	0.93
200	0.456	0.364	0.96
250	0.441	0.360	0.93
300	0.435	0.357	1.00
350	0.428	0.354	1.00
400	0.418	0.355	1.00
450	0.411	0.355	1.00
500	0.397	0.353	1.00
550	0.393	0.352	1.00
600	0.388	0.354	1.00
650	0.384	0.352	1.00
700	0.387	0.349	1.00
750	0.384	0.348	1.00
800	0.382	0.345	1.00
850	0.381	0.343	1.00
900	0.382	0.342	1.00
950	0.380	0.343	1.00
1000	0.376	0.345	1.00

(c) Evaluation on $H_1^{\ominus}SE(2)$.

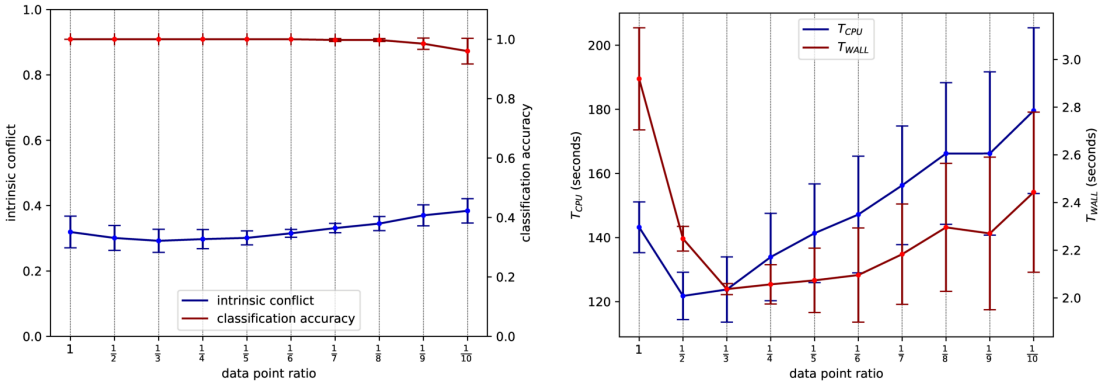
Timestep	$\Delta[\Xi]/\sum I$		Accuracy
-	μ	σ	-
50	0.570	0.348	0.49
100	0.525	0.333	0.80
150	0.508	0.328	0.91
200	0.474	0.326	0.89
250	0.432	0.330	0.93
300	0.415	0.329	0.93
350	0.414	0.325	0.93
400	0.414	0.321	0.98
450	0.405	0.319	0.96
500	0.386	0.314	1.00
550	0.379	0.312	1.00
600	0.372	0.311	1.00
650	0.363	0.309	1.00
700	0.357	0.309	1.00
750	0.346	0.310	1.00
800	0.340	0.310	1.00
850	0.339	0.307	1.00
900	0.334	0.308	1.00
950	0.333	0.306	1.00
1000	0.331	0.305	1.00

(d) Evaluation on $H_2^{\ominus}SE(2)$.

Table 7.17: Results for Scenario V, intermediate conflict calculation. Evaluation on hypothesis sets $H_1^{\oplus}SE(2)$, $H_2^{\oplus}SE(2)$, $H_1^{\ominus}SE(2)$ and $H_2^{\ominus}SE(2)$ for $t = 50$ to $t = 1000$. Intrinsic conflict (μ and σ) and classification accuracy.

num_iterations	5
dim_controls	3 and 2
noise_actuator	0.0
enable_sensors	true
noise_sensors	0.0
agent_stepsize	1.5 (Scenario II ,Section 7.4.2)
exploration_pattern	$C[SA]$ (Scenario II ,Section 7.4.2)
conflict_function	Δ_{NW}
conflict_sigma_factor	0.3 (Scenario I ,Section 7.4.1)
conflict_subsets	$(1, 1.0), (2, \frac{1}{2}), (3, \frac{1}{3}), (4, \frac{1}{4}), (5, \frac{1}{5}), (6, \frac{1}{6}), (7, \frac{1}{7}), (8, \frac{1}{8}), (9, \frac{1}{9})$ and $(10, \frac{1}{10})$
bootstrapping_type	classify
pso_num_particles	–
pso_parameterization	–
hypothesis_group	$SE(2)$
hypothesis_set	$\mathrm{H}_1^{\oplus}\mathrm{SE}(2)$, $\mathrm{H}_2^{\oplus}\mathrm{SE}(2)$, $\mathrm{H}_1^{\ominus}\mathrm{SE}(2)$, and $\mathrm{H}_2^{\ominus}\mathrm{SE}(2)$

Table 7.18: Parameters for Scenario V, subset conflict calculation.



(a) Intrinsic conflict and classification accuracy for $(n = 1, r = 1.0)$ to $(n = 10, r = 0.1)$.

(b) Computation time (cpu ticks and wall clock time) for $(n = 1, r = 1.0)$ to $(n = 10, r = 0.1)$.

Figure 7.13: Results from Scenario V, subset conflict calculation. Intrinsic conflict, classification accuracy and timing data for $(n = 1, r = 1.0)$ to $(n = 10, r = 0.1)$. Mean and standard deviation over all results for hypothesis sets $\mathrm{H}_1^{\oplus}\mathrm{SE}(2)$, $\mathrm{H}_2^{\oplus}\mathrm{SE}(2)$, $\mathrm{H}_1^{\ominus}\mathrm{SE}(2)$, and $\mathrm{H}_2^{\ominus}\mathrm{SE}(2)$.

Subsets	Ratio	$\Delta[\Xi]/\sum I$		Accuracy
n	r	μ	σ	-
1	1.000	0.259	0.242	1.00
2	0.500	0.245	0.241	1.00
3	0.333	0.241	0.243	1.00
4	0.250	0.255	0.243	1.00
5	0.200	0.270	0.245	1.00
6	0.166	0.308	0.250	1.00
7	0.142	0.342	0.255	0.99
8	0.125	0.366	0.260	0.99
9	0.111	0.404	0.274	0.96
10	0.100	0.424	0.280	0.90

(a) Evaluation on $H_1^{\oplus}SE(2)$.

Subsets	Ratio	$\Delta[\Xi]/\sum I$		Accuracy
n	r	μ	σ	-
1	1.000	0.311	0.364	1.00
2	0.500	0.320	0.363	1.00
3	0.333	0.320	0.365	1.00
4	0.250	0.320	0.365	1.00
5	0.200	0.312	0.368	1.00
6	0.166	0.311	0.369	1.00
7	0.142	0.325	0.365	1.00
8	0.125	0.342	0.363	1.00
9	0.111	0.361	0.365	1.00
10	0.100	0.369	0.365	1.00

(b) Evaluation on $H_2^{\oplus}SE(2)$.

Subsets	Ratio	$\Delta[\Xi]/\sum I$		Accuracy
n	r	μ	σ	-
1	1.000	0.376	0.345	1.00
2	0.500	0.329	0.348	1.00
3	0.333	0.310	0.351	1.00
4	0.250	0.314	0.349	1.00
5	0.200	0.318	0.348	1.00
6	0.166	0.333	0.349	1.00
7	0.142	0.344	0.349	1.00
8	0.125	0.356	0.352	1.00
9	0.111	0.386	0.349	1.00
10	0.100	0.403	0.348	0.96

(c) Evaluation on $H_1^{\ominus}SE(2)$.

Subsets	Ratio	$\Delta[\Xi]/\sum I$		Accuracy
n	r	μ	σ	-
1	1.000	0.331	0.305	1.00
2	0.500	0.310	0.306	1.00
3	0.333	0.298	0.310	1.00
4	0.250	0.301	0.309	1.00
5	0.200	0.305	0.308	1.00
6	0.166	0.309	0.307	1.00
7	0.142	0.314	0.307	1.00
8	0.125	0.316	0.308	1.00
9	0.111	0.330	0.308	0.98
10	0.100	0.340	0.308	0.98

(d) Evaluation on $H_2^{\ominus}SE(2)$.

Table 7.19: Results for Scenario V, subset conflict calculation. Intrinsic conflict (μ and σ) and classification accuracy. Evaluation on hypothesis sets $H_1^{\oplus}SE(2)$, $H_2^{\oplus}SE(2)$, $H_1^{\ominus}SE(2)$ and $H_2^{\ominus}SE(2)$ for $(n = 1, r = 1.0)$ to $(n = 10, r = 0.1)$.

$H_1^{\oplus}SE(2)$ and $n = 10$, as given in Table 7.19a. For $H_2^{\oplus}SE(2)$, it remains constant at 1.0 for all cases as depicted in Table 7.19b.

For almost all cases, the minimum mean intrinsic conflict is given for $n = 3$, i.e. the output of AgentLoop::Classify seems to be slightly more robust when utilizing $n = 3$ subsets with $r = 0.333$ percent of the data points each. However, $H_2^{\oplus}SE(2)$ given in Table 7.19b is as an exception as for $n = 2$ to $n = 5$ the mean intrinsic conflict is higher than for $n = 1$ and $n = 6$.

We now consider the computation time T_{CPU} and T_{WALL} as given in Table 7.20. The former provides us the computation time, summed over all individual CPU cores, the latter with the time passed as seen on a regular clock. As the conflict calculation makes use of the maximum number of threads available (in our case 88), T_{CPU} will always be larger than T_{WALL}, as the former counts the computation time of multiple threads working in parallel. The computation time for a single-core CPU corresponds (approximately) to T_{CPU}, as this value denotes the total amount of processor time.

In most cases the minimum of T_{CPU} is given at $n = 2$. Only for $H_2^{\ominus}SE(2)$ in Table 7.20d the minimum is given at $n = 3$. The minimum of T_{WALL} is given at $n = 3$ for $H_1^{\oplus}SE(2)$ and $H_2^{\oplus}SE(2)$ as depicted in Tables 7.20a and 7.20b respectively. For $H_1^{\ominus}SE(2)$ and $H_2^{\ominus}SE(2)$ given in Tables 7.20c and 7.20d the minimum for T_{WALL} is associated to $n = 4$ and $n = 6$ respectively.

Subsets	Ratio	T_{CPU}		T_{WALL}	
n	r	μ	σ	μ	σ
1	1.000	133.132	1.987	2.623	0.062
2	0.500	132.511	1.882	2.186	0.039
3	0.333	138.472	2.197	2.062	0.042
4	0.250	153.238	3.016	2.177	0.041
5	0.200	163.276	1.936	2.262	0.018
6	0.166	173.739	6.208	2.381	0.092
7	0.142	182.913	5.440	2.484	0.074
8	0.125	197.826	5.062	2.671	0.078
9	0.111	202.923	10.040	2.726	0.145
10	0.100	216.174	6.378	2.903	0.087

(a) Evaluation on $\mathrm{H}_1^{\oplus}\mathrm{SE}(2)$.

Subsets	Ratio	T_{CPU}		T_{WALL}	
n	r	μ	σ	μ	σ
1	1.000	140.806	5.755	2.906	0.120
2	0.500	120.689	3.460	2.225	0.075
3	0.333	122.904	2.251	2.010	0.054
4	0.250	133.838	5.959	2.037	0.096
5	0.200	140.438	4.775	2.037	0.061
6	0.166	143.787	6.462	2.012	0.088
7	0.142	153.953	6.723	2.120	0.087
8	0.125	162.840	13.045	2.229	0.186
9	0.111	161.614	14.986	2.180	0.224
10	0.100	173.990	6.522	2.348	0.097

(b) Evaluation on $\mathrm{H}_2^{\oplus}\mathrm{SE}(2)$.

Subsets	Ratio	T_{CPU}		T_{WALL}	
n	r	μ	σ	μ	σ
1	1.000	148.437	2.902	3.038	0.062
2	0.500	116.370	3.495	2.286	0.051
3	0.333	116.738	4.105	2.049	0.032
4	0.250	124.364	7.576	2.023	0.092
5	0.200	133.042	7.642	2.053	0.095
6	0.166	137.938	7.081	2.059	0.090
7	0.142	147.173	8.850	2.136	0.109
8	0.125	157.324	7.700	2.249	0.114
9	0.111	155.798	12.401	2.199	0.186
10	0.100	172.959	15.167	2.421	0.218

(c) Evaluation on $\mathrm{H}_1^{\ominus}\mathrm{SE}(2)$.

Subsets	Ratio	T_{CPU}		T_{WALL}	
n	r	μ	σ	μ	σ
1	1.000	150.489	9.778	3.108	0.162
2	0.500	117.589	5.579	2.295	0.121
3	0.333	117.107	5.588	2.026	0.095
4	0.250	124.260	6.884	1.990	0.097
5	0.200	128.661	4.480	1.942	0.069
6	0.166	133.308	5.258	1.931	0.071
7	0.142	141.133	7.713	1.993	0.118
8	0.125	146.890	5.970	2.034	0.096
9	0.111	144.590	4.930	1.976	0.067
10	0.100	155.238	2.237	2.101	0.028

(d) Evaluation on $\mathrm{H}_2^{\ominus}\mathrm{SE}(2)$.

Table 7.20: Results for Scenario V, subset conflict calculation. Timing data. Evaluation on hypothesis sets $\mathrm{H}_1^{\oplus}\mathrm{SE}(2)$, $\mathrm{H}_2^{\oplus}\mathrm{SE}(2)$, $\mathrm{H}_1^{\ominus}\mathrm{SE}(2)$ and $\mathrm{H}_2^{\ominus}\mathrm{SE}(2)$ for $(n = 1, r = 1.0)$ to $(n = 10, r = 0.1)$. Timing data.

Figures 7.13a and 7.13b show the mean and standard deviation for the intrinsic conflict, T_{CPU} and T_{WALL} over all cases. Taking into account the classification accuracy, the mean intrinsic conflict and both T_{CPU} and T_{WALL}, either $n = 2$ or $n = 3$ seem to be a good compromise between accuracy, robustness and computation time.

7.4.5.3 Sensor Noise Evaluation

In the third scenario, we investigate the effect actuator and sensor noise has on the performance of AgentLoop::Classify. We will initially look at the effect noise has when applied to the actuator and the sensors individually. We then will add noise to both sensors and actuator simultaneously.

Parameters We initialize the scenario with the parameters given in Table 7.21.

Results The results are depicted in Table 7.22 and Fig. 7.14 with additional figures and tables given in Appendices D.8.3 and D.9.3. In both the holonomic and the nonholonomic case the classification accuracy remains relatively stable up until $\sigma_p = 0.25$. For the holonomic case given in Tables 7.22a and 7.22b the classification accuracy for $\sigma_p = 0.50$ only drops slightly to 0.99. For the nonholonomic case given in Tables 7.22c and 7.22d for the same noise level it drops

num_iterations	5
dim_controls	3 and 2
noise_actuator	0.0
enable_sensors	true
noise_sensors	0.001, 0.0025, 0.005, 0.01, 0.025, 0.05, 0.1, 0.25, 0.5, 0.75 and 1.0
agent_stepsize	1.5 (Scenario II ,Section 7.4.2)
exploration_pattern	$C[SA]$ (Scenario II ,Section 7.4.2)
conflict_function	Δ_{NW}
conflict_sigma_factor	0.3 (Scenario I ,Section 7.4.1)
conflict_subsets	(1, 1.0)
bootstrapping_type	classify
pso_num_particles	–
pso_parameterization	–
hypothesis_group	$SE(2)$
hypothesis_set	$H_1^{\oplus}SE(2)$, $H_2^{\oplus}SE(2)$, $H_1^{\ominus}SE(2)$ and $H_2^{\ominus}SE(2)$

Table 7.21: Parameters for Scenario V, sensor noise evaluation.

Noise	$\Delta[\Xi]/\sum I$		Accuracy
σ_p	μ	σ	-
0.001	0.259	0.242	1.00
0.003	0.260	0.242	1.00
0.005	0.260	0.242	1.00
0.010	0.260	0.242	1.00
0.025	0.260	0.242	1.00
0.050	0.260	0.242	1.00
0.100	0.257	0.241	1.00
0.250	0.248	0.240	1.00
0.500	0.310	0.258	0.99
0.750	0.389	0.287	0.81
1.000	0.422	0.295	0.61

(a) Evaluation on $H_1^{\oplus}SE(2)$

Noise	$\Delta[\Xi]/\sum I$		Accuracy
σ_p	μ	σ	-
0.001	0.311	0.364	1.00
0.003	0.311	0.364	1.00
0.005	0.311	0.364	1.00
0.010	0.311	0.364	1.00
0.025	0.311	0.365	1.00
0.050	0.311	0.365	1.00
0.100	0.311	0.365	1.00
0.250	0.316	0.366	1.00
0.500	0.362	0.367	1.00
0.750	0.425	0.387	0.92
1.000	0.453	0.392	0.76

(b) Evaluation on $H_2^{\oplus}SE(2)$

Noise	$\Delta[\Xi]/\sum I$		Accuracy
σ_p	μ	σ	-
0.001	0.376	0.345	1.00
0.003	0.376	0.345	1.00
0.005	0.376	0.345	1.00
0.010	0.376	0.345	1.00
0.025	0.375	0.345	1.00
0.050	0.371	0.345	1.00
0.100	0.358	0.347	1.00
0.250	0.318	0.351	1.00
0.500	0.437	0.351	0.80
0.750	0.506	0.354	0.60
1.000	0.525	0.353	0.53

(c) Evaluation on $H_1^{\ominus}SE(2)$

Noise	$\Delta[\Xi]/\sum I$		Accuracy
σ_p	μ	σ	-
0.001	0.331	0.305	1.00
0.003	0.331	0.305	1.00
0.005	0.331	0.305	1.00
0.010	0.331	0.305	1.00
0.025	0.331	0.305	1.00
0.050	0.330	0.305	1.00
0.100	0.326	0.305	1.00
0.250	0.321	0.305	1.00
0.500	0.369	0.324	0.93
0.750	0.432	0.342	0.58
1.000	0.438	0.339	0.49

(d) Evaluation on $H_2^{\ominus}SE(2)$

Table 7.22: Results for Scenario V, sensor noise evaluation. Evaluation on hypothesis sets $H_1^{\oplus}SE(2)$, $H_2^{\oplus}SE(2)$, $H_1^{\ominus}SE(2)$ and $H_2^{\ominus}SE(2)$. Intrinsic conflict (μ and σ) and classification accuracy.

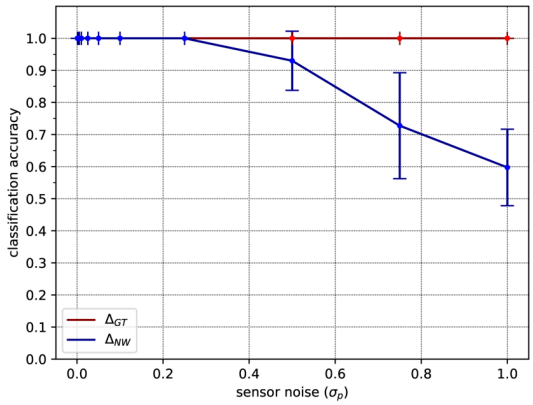


Figure 7.14: Results from Scenario V, sensor noise evaluation. Comparison of classification accuracy between Scenario IV and Scenario V. Mean and standard deviation over all results for hypothesis sets $H_1^{\oplus}SE(2)$, $H_2^{\oplus}SE(2)$, $H_1^{\ominus}SE(2)$, and $H_2^{\ominus}SE(2)$.

num_iterations	5
dim_controls	3 and 2
noise_actuator	0.001, 0.0025, 0.005, 0.01, 0.025, 0.05, 0.1, 0.25, 0.5, 0.75 and 1.0
enable_sensors	true
noise_sensors	0.0
agent_stepsize	1.5 (Scenario II ,Section 7.4.2)
exploration_pattern	$C[SA]$ (Scenario II ,Section 7.4.2)
conflict_function	Δ_{NW}
conflict_sigma_factor	0.3 (Scenario I ,Section 7.4.1)
conflict_subsets	(1, 1.0)
bootstrapping_type	classify
pso_num_particles	–
pso_parameterization	–
hypothesis_group	$SE(2)$
hypothesis_set	$H_1^{\oplus}SE(2)$, $H_2^{\oplus}SE(2)$, $H_1^{\ominus}SE(2)$ and $H_2^{\ominus}SE(2)$

Table 7.23: Parameters for Scenario V, actuator noise evaluation.

to 0.80 and 0.93 respectively. Comparing these results to the sensor noise evaluation in Scenario IV (see Section 7.4.4.2), we see that AgentLoop::Classify with Δ_{NW} is more susceptible to sensor noise. While the classification accuracy in Scenario IV never drops below 1.0, in this setting it does, reaching the minimum if 0.49 at $\sigma_p = 1.0$.

7.4.5.4 Actuator Noise Evaluation

Parameters We initialize the scenario with the parameters given in Table 7.23.

Results The results are depicted in Table 7.24 and Fig. 7.15 with additional figures and tables given in Appendices D.8.4 and D.9.4. The classification accuracy remains stable up until $\sigma_c = 0.05$ for almost all cases. Only for $H_2^{\ominus}SE(2)$, as given in Table 7.24d, it drops below 1.0 for $\sigma_c = 0.025$ already. Increasing the noise further causes the classification accuracy to drop

Noise	$\Delta[\Xi]/\sum I$		Accuracy
σ_c	μ	σ	-
0.001	0.254	0.240	1.00
0.003	0.256	0.244	1.00
0.005	0.263	0.242	1.00
0.010	0.277	0.246	1.00
0.025	0.312	0.251	1.00
0.050	0.351	0.263	1.00
0.100	0.406	0.282	0.91
0.250	0.472	0.296	0.59
0.500	0.500	0.295	0.26
0.750	0.500	0.292	0.13
1.000	0.498	0.297	0.09

(a) Evaluation on $H_1^{\oplus}SE(2)$.

Noise	$\Delta[\Xi]/\sum I$		Accuracy
σ_c	μ	σ	-
0.001	0.330	0.362	1.00
0.003	0.323	0.360	1.00
0.005	0.316	0.363	1.00
0.010	0.330	0.365	1.00
0.025	0.349	0.362	1.00
0.050	0.416	0.371	1.00
0.100	0.432	0.370	0.92
0.250	0.522	0.384	0.40
0.500	0.518	0.381	0.28
0.750	0.500	0.374	0.16
1.000	0.502	0.380	0.28

(b) Evaluation on $H_2^{\oplus}SE(2)$.

Noise	$\Delta[\Xi]/\sum I$		Accuracy
σ_c	μ	σ	-
0.001	0.375	0.351	1.00
0.003	0.373	0.349	1.00
0.005	0.380	0.347	1.00
0.010	0.389	0.350	1.00
0.025	0.404	0.349	1.00
0.050	0.425	0.357	1.00
0.100	0.441	0.365	0.82
0.250	0.483	0.356	0.36
0.500	0.426	0.364	0.18
0.750	0.455	0.354	0.13
1.000	0.426	0.349	0.16

(c) Evaluation on $H_1^{\ominus}SE(2)$.

Noise	$\Delta[\Xi]/\sum I$		Accuracy
σ_c	μ	σ	-
0.001	0.324	0.305	1.00
0.003	0.335	0.307	1.00
0.005	0.347	0.309	1.00
0.010	0.355	0.310	1.00
0.025	0.379	0.317	0.93
0.050	0.392	0.331	0.91
0.100	0.460	0.339	0.80
0.250	0.527	0.335	0.58
0.500	0.553	0.323	0.44
0.750	0.552	0.327	0.24
1.000	0.513	0.323	0.33

(d) Evaluation on $H_2^{\ominus}SE(2)$.

Table 7.24: Results for Scenario V, actuator noise evaluation. Evaluation on hypothesis sets $H_1^{\oplus}SE(2)$, $H_2^{\oplus}SE(2)$, $H_1^{\ominus}SE(2)$ and $H_2^{\ominus}SE(2)$. Intrinsic conflict (μ and σ) and classification accuracy.

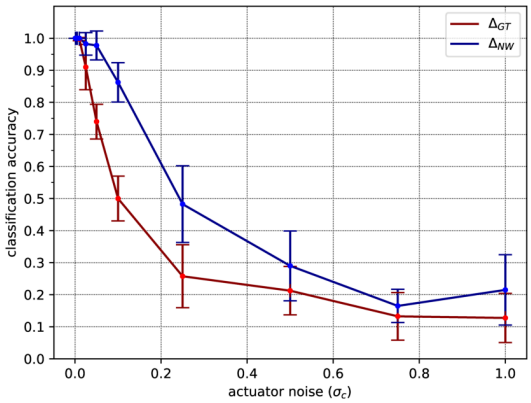


Figure 7.15: Results from Scenario V, actuator noise evaluation. Comparison of classification accuracy between Scenario IV and Scenario V. Mean and standard deviation over all results for hypothesis sets $H_1^{\oplus}SE(2)$, $H_2^{\oplus}SE(2)$, $H_1^{\ominus}SE(2)$, and $H_2^{\ominus}SE(2)$.

num_iterations	5
dim_controls	3 and 2
noise_actuator	0.001, 0.0025, 0.005, 0.01, 0.025, 0.05, 0.1, 0.25, 0.5, 0.75 and 1.0
enable_sensors	true
noise_sensors	sensor noise (all sensors) matches actuator noise
agent_stepsize	1.5 (Scenario II ,Section 7.4.2)
exploration_pattern	$C[SA]$ (Scenario II ,Section 7.4.2)
conflict_function	Δ_{NW}
conflict_sigma_factor	0.3 (Scenario I ,Section 7.4.1)
conflict_subsets	(1, 1.0)
bootstrapping_type	classify
pso_num_particles	–
pso_parameterization	–
hypothesis_group	$SE(2)$
hypothesis_set	$\mathrm{H}_1^{\oplus}\mathrm{SE}(2)$, $\mathrm{H}_2^{\oplus}\mathrm{SE}(2)$, $\mathrm{H}_1^{\ominus}\mathrm{SE}(2)$ and $\mathrm{H}_2^{\ominus}\mathrm{SE}(2)$

Table 7.25: Parameters for Scenario V, actuator and sensor noise evaluation.

rapidly. In all cases it drops to approximately chance level accuracy for the individual hypothesis sets at at $\sigma_c = 0.75$ or $\sigma_c = 1.0$, given as $\frac{1}{14} = 0.07$ for $\mathrm{H}_1^{\oplus}\mathrm{SE}(2)$, $\frac{1}{5} = 0.25$ for $\mathrm{H}_1^{\oplus}\mathrm{SE}(2)$ and $\frac{1}{9} = 0.11$ for both $\mathrm{H}_1^{\ominus}\mathrm{SE}(2)$ and $\mathrm{H}_2^{\ominus}\mathrm{SE}(2)$. The only exception is $\mathrm{H}_2^{\ominus}\mathrm{SE}(2)$ at Table 7.24d which only drops to a minimum classification accuracy of 0.24 at $\sigma_c = 0.75$. Comparing these results to the corresponding case from Scenario IVa (see Section 7.4.4), we see differences in the noise levels that cause the classification accuracy to drop. In Scenario IV the classification accuracy remained stable up until $\sigma_p = 0.01$ to $\sigma_p = 0.025$. In this scenario the classification accuracy remains stable up until $\sigma_c = 0.05$ with only the previously mentioned exception of $\sigma_c = 0.01$ for $\mathrm{H}_2^{\ominus}\mathrm{SE}(2)$ while even in these cases it only drops to 0.9. The decline in classification accuracy for AgentLoop::Classify using Δ_{NW} compared to $\Delta_{\mathrm{GT}}[\Omega_t]$ starts only at a noise level two to four times as high.

7.4.5.5 Actuator and Sensor Noise Evaluation

Parameters We initialize the scenario with the parameters given in Table 7.25.

Results The results are depicted in Table 7.26 and Fig. 7.16 with additional figures and tables given in Appendices D.8.5 and D.9.5. As in the previous experiment, the classification accuracy remains stable up until $\sigma_c, \sigma_p = 0.05$ for almost all cases. Only for $\mathrm{H}_2^{\ominus}\mathrm{SE}(2)$, as given in Table 7.26d, it drops below 1.0 for $\sigma_c, \sigma_p = 0.025$ already. Increasing the noise further causes the classification accuracy to drop rapidly. In almost all cases it stays above chance level accuracy for the individual hypothesis sets. The only exception is $\mathrm{H}_2^{\oplus}\mathrm{SE}(2)$ in Table 7.26b which drops substantially lower than $\frac{1}{5} = 0.25$, reaching a minimum classification accuracy of 0.04 for $\sigma_c, \sigma_p = 1.0$. Comparing these results to the corresponding case from Scenario IV (see Section 7.4.4), we see differences in the noise levels that cause the classification accuracy to drop. In Scenario IVa the classification accuracy remained stable up until $\sigma_c, \sigma_p = 0.01$ to $\sigma_c, \sigma_p = 0.025$. Here the classification accuracy remains stable up until the $\sigma_c, \sigma_p = 0.05$ with only the exception of $\sigma_c, \sigma_p = 0.01$ for $\mathrm{H}_2^{\ominus}\mathrm{SE}(2)$. Like observed in the previous experiment

Noise	$\Delta[\Xi]/\sum I$		Accuracy
σ_c, σ_p	μ	σ	-
0.001	0.254	0.240	1.00
0.003	0.256	0.244	1.00
0.005	0.263	0.242	1.00
0.010	0.277	0.246	1.00
0.025	0.311	0.251	1.00
0.050	0.349	0.263	1.00
0.100	0.394	0.279	0.94
0.250	0.473	0.294	0.64
0.500	0.514	0.290	0.24
0.750	0.489	0.290	0.14
1.000	0.489	0.299	0.13

(a) Evaluation on $H_1^{\oplus}SE(2)$.

Noise	$\Delta[\Xi]/\sum I$		Accuracy
σ_c, σ_p	μ	σ	-
0.001	0.330	0.362	1.00
0.003	0.323	0.360	1.00
0.005	0.316	0.363	1.00
0.010	0.330	0.365	1.00
0.025	0.347	0.363	1.00
0.050	0.415	0.370	1.00
0.100	0.428	0.369	0.92
0.250	0.533	0.376	0.52
0.500	0.513	0.373	0.28
0.750	0.512	0.383	0.08
1.000	0.502	0.369	0.04

(b) Evaluation on $H_2^{\oplus}SE(2)$.

Noise	$\Delta[\Xi]/\sum I$		Accuracy
σ_c, σ_p	μ	σ	-
0.001	0.375	0.351	1.00
0.003	0.373	0.349	1.00
0.005	0.380	0.347	1.00
0.010	0.389	0.350	1.00
0.025	0.403	0.349	1.00
0.050	0.421	0.357	1.00
0.100	0.429	0.365	0.91
0.250	0.456	0.360	0.73
0.500	0.514	0.349	0.47
0.750	0.561	0.355	0.31
1.000	0.537	0.355	0.31

(c) Evaluation on $H_1^{\ominus}SE(2)$.

Noise	$\Delta[\Xi]/\sum I$		Accuracy
σ_c, σ_p	μ	σ	-
0.001	0.324	0.305	1.00
0.003	0.335	0.307	1.00
0.005	0.347	0.309	1.00
0.010	0.355	0.310	1.00
0.025	0.378	0.317	0.93
0.050	0.393	0.332	0.91
0.100	0.455	0.339	0.84
0.250	0.479	0.339	0.60
0.500	0.472	0.328	0.22
0.750	0.441	0.329	0.29
1.000	0.438	0.321	0.29

(d) Evaluation on $H_2^{\ominus}SE(2)$.

Table 7.26: Results for Scenario V, actuator and sensor noise evaluation. Evaluation on hypothesis sets $H_1^{\oplus}SE(2)$, $H_2^{\oplus}SE(2)$, $H_1^{\ominus}SE(2)$ and $H_2^{\ominus}SE(2)$. Intrinsic conflict (μ and σ) and classification accuracy.

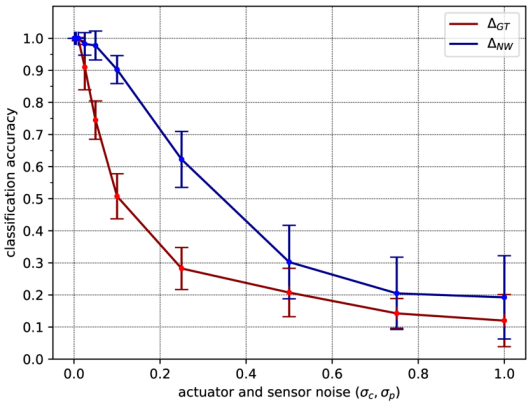


Figure 7.16: Results from Scenario V, actuator and sensor noise evaluation. Comparison of classification accuracy between Scenario IV and Scenario V. Mean and standard deviation over all results for hypothesis sets $H_1^{\oplus}SE(2)$, $H_2^{\oplus}SE(2)$, $H_1^{\ominus}SE(2)$, and $H_2^{\ominus}SE(2)$.

num_iterations	5
dim_controls	3 and 2
noise_actuator	0.0
enable_sensors	all 16 sensor permutations of p_L, p_P, p_R and p_T, given as $p_0 = (p_L[\bullet], p_P[\bullet], p_R[\bullet], p_T[\bullet])$, $p_1 = (p_L[\bullet], p_P[\bullet], p_R[\bullet], p_T[\circ])$, $p_2 = (p_L[\bullet], p_P[\bullet], p_R[\circ], p_T[\bullet])$, $p_3 = (p_L[\bullet], p_P[\bullet], p_R[\circ], p_T[\circ])$ ... $p_{15} = (p_L[\circ], p_P[\circ], p_R[\circ], p_T[\circ])$
noise_sensors	0.0
agent_stepsize	1.5 (Scenario II ,Section 7.4.2)
exploration_pattern	$C[SA]$ (Scenario II ,Section 7.4.2)
conflict_function	Δ_{NW}
conflict_sigma_factor	0.3 (Scenario I ,Section 7.4.1)
conflict_subsets	(1, 1.0)
bootstrapping_type	classify
pso_num_particles	–
pso_parameterization	–
hypothesis_group	$SE(2)$
hypothesis_set	$\text{H}_1^{\oplus}\text{SE}(2)$, $\text{H}_2^{\oplus}\text{SE}(2)$, $\text{H}_1^{\ominus}\text{SE}(2)$ and $\text{H}_2^{\ominus}\text{SE}(2)$

Table 7.27: Parameters for Scenario V, sensor permutations with occupancy.

the decline in classification accuracy for AgentLoop::Classify using Δ_{NW} compared to $\Delta_{\text{GT}}[\Omega_t]$ starts only at a noise level two to four times as high.

7.4.5.6 Sensor Permutations With Occupancy

In the fourth experiment we will investigate the impact individual sensors have on the classification accuracy. As given in Section 7.2.2 we selected the sensors based on the different ways that their perceptions are related to the geometry of the configuration space. Thus we are interested in evaluating whether this geometric difference has an impact on the classification accuracy. To that end we created a number of 16 agents, each equipped with an individual permutation of sensors. Each of the agents was selected as a prototype for the usual experimental setup as described in Section 7.2.5.2.

Parameters We initialize the scenario with the parameters given in Table 7.27.

Results The results are depicted in Table 7.28 with additional figures and tables given in Appendices D.8.6 and D.9.6. We can see that different combinations of sensors don't have a substantial impact on the classification accuracy. In the holonomic case it only drops below 1.0 in two cases as depicted in Tables 7.28a and 7.28b. In the nonholonomic case given in Tables 7.28c and 7.28d the classification accuracy drops lower, with a minimum of 0.87 for $\text{H}_2^{\ominus}\text{SE}(2)$ with only the proximity sensor $p_P[\bullet]$ enabled.

Notably even for an agent equipped with zero sensors, the classification accuracy only drops to a minimum of 0.8. This is due to the fact that Δ_{NW} calculates the conflict based on two distinct types of data points: samples that are composed from the perceptions of the individual sensors, and occupancy information that reflects the feedback the agent receives when encountering an

Sensors				$\Delta[\Xi]/\sum I$		Accuracy
p_L	p_P	p_R	p_T	μ	σ	-
○	○	○	○	0.397	0.273	0.91
○	○	○	●	0.313	0.249	1.00
○	○	●	○	0.376	0.266	1.00
○	○	●	●	0.326	0.253	1.00
○	●	○	○	0.354	0.265	1.00
○	●	○	●	0.308	0.248	1.00
○	●	●	○	0.346	0.257	1.00
○	●	●	●	0.317	0.248	1.00
●	○	○	○	0.355	0.260	1.00
●	○	○	●	0.314	0.250	1.00
●	○	●	○	0.360	0.265	1.00
●	○	●	●	0.327	0.256	1.00
●	●	○	○	0.330	0.255	1.00
●	●	○	●	0.303	0.249	1.00
●	●	●	○	0.337	0.256	1.00
●	●	●	●	0.314	0.251	1.00

(a) Evaluation on $\mathrm{H}_1^{\oplus}\mathrm{SE}(2)$.

Sensors				$\Delta[\Xi]/\sum I$		Accuracy
p_L	p_P	p_R	p_T	μ	σ	-
○	○	○	○	0.441	0.375	0.92
○	○	○	●	0.364	0.361	1.00
○	○	●	○	0.400	0.365	1.00
○	○	●	●	0.363	0.361	1.00
○	●	○	○	0.422	0.372	0.96
○	●	○	●	0.369	0.363	1.00
○	●	●	○	0.398	0.365	1.00
○	●	●	●	0.364	0.364	1.00
●	○	○	○	0.394	0.365	1.00
●	○	○	●	0.369	0.358	1.00
●	○	●	○	0.385	0.362	1.00
●	○	●	●	0.368	0.359	1.00
●	●	○	○	0.393	0.364	1.00
●	●	○	●	0.370	0.360	1.00
●	●	●	○	0.386	0.361	1.00
●	●	●	●	0.369	0.359	1.00

(b) Evaluation on $\mathrm{H}_2^{\oplus}\mathrm{SE}(2)$.

Sensors				$\Delta[\Xi]/\sum I$		Accuracy
p_L	p_P	p_R	p_T	μ	σ	-
○	○	○	○	0.466	0.362	0.80
○	○	○	●	0.402	0.356	1.00
○	○	●	○	0.468	0.348	0.93
○	○	●	●	0.412	0.349	1.00
○	●	○	○	0.428	0.361	0.96
○	●	○	●	0.399	0.360	1.00
○	●	●	○	0.425	0.354	0.98
○	●	●	●	0.400	0.355	1.00
●	○	○	○	0.431	0.349	1.00
●	○	○	●	0.388	0.354	1.00
●	○	●	○	0.458	0.340	0.98
●	○	●	●	0.409	0.348	1.00
●	●	○	○	0.412	0.351	1.00
●	●	○	●	0.387	0.356	1.00
●	●	●	○	0.424	0.348	1.00
●	●	●	●	0.397	0.353	1.00

(c) Evaluation on $\mathrm{H}_1^{\ominus}\mathrm{SE}(2)$.

Sensors				$\Delta[\Xi]/\sum I$		Accuracy
p_L	p_P	p_R	p_T	μ	σ	-
○	○	○	○	0.449	0.314	0.89
○	○	○	●	0.398	0.309	0.98
○	○	●	○	0.410	0.319	0.96
○	○	●	●	0.388	0.312	1.00
○	●	○	○	0.436	0.318	0.87
○	●	○	●	0.408	0.312	0.98
○	●	●	○	0.415	0.317	0.98
○	●	●	●	0.399	0.312	1.00
●	○	○	○	0.405	0.317	0.98
●	○	○	●	0.382	0.313	0.98
●	○	●	○	0.392	0.321	0.98
●	○	●	●	0.377	0.315	1.00
●	●	○	○	0.406	0.320	0.96
●	●	○	●	0.391	0.314	0.98
●	●	●	○	0.396	0.318	0.98
●	●	●	●	0.386	0.314	1.00

(d) Evaluation on $\mathrm{H}_2^{\ominus}\mathrm{SE}(2)$.

Table 7.28: Results for Scenario V, sensor permutations with occupancy. Evaluation on hypothesis sets $\mathrm{H}_1^{\oplus}\mathrm{SE}(2)$, $\mathrm{H}_2^{\oplus}\mathrm{SE}(2)$ $\mathrm{H}_1^{\ominus}\mathrm{SE}(2)$ and $\mathrm{H}_2^{\ominus}\mathrm{SE}(2)$. Occupancy data enabled with $p_O[\bullet]$. Intrinsic conflict (μ and σ) and classification accuracy.

obstacle in its environment. Obviously occupancy data is sufficient for discovering the true hypothesis reasonably well. While this is an interesting result on its own, it undermines our aim of the determining the impact individual sensors measurements have on the classification accuracy.

7.4.5.7 Sensor Permutations Without Occupancy

To that end, we modify the scenario by disabling the utilization of occupancy data p_O in Δ_{NW}, i.e. we disable the term in the conflict function Δ_{NW} (see Eq. (6.30)) that calculates and incorpo-

num_iterations	100
dim_controls	3 and 2
noise_actuator	0.0
enable_sensors	all 16 sensor permutations as given in Table 7.27 utilization of occupancy data disabled with $p_O[\circ]$
noise_sensors	0.0
agent_stepsize	1.5 (Scenario II ,Section 7.4.2)
exploration_pattern	$C[SA]_{[50]}$ consisting of the first 50 controls of $C[SA]$ (Scenario II ,Section 7.4.2)
conflict_function	Δ_{NW}
conflict_sigma_factor	0.3 (Scenario I ,Section 7.4.1)
conflict_subsets	(1, 1.0)
bootstrapping_type	classify
pso_num_particles	–
pso_parameterization	–
hypothesis_group	$SE(2)$
hypothesis_set	$H_1^{\oplus}SE(2)$, $H_2^{\oplus}SE(2)$, $H_1^{\ominus}SE(2)$ and $H_2^{\ominus}SE(2)$

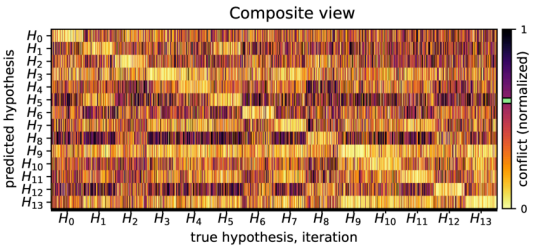
Table 7.29: Parameters for Scenario V, sensor permutations without occupancy.

rates conflict on the sensorimotor obstruction sets i.e., the occupancy data. Thus we deprive the agent of the ability to utilize information on detected obstacles in AgentLoop::Classify. This change however only affects the way available data is used and not the physical interaction between the agent and the environment, i.e. the agent still collides with obstacles. We performed an intermediate evaluation which showed that disabling occupancy data usage was not sufficient to see a notable difference between sensor permutations as the result of AgentLoop::Classify was still too good, i.e. too similar for all permutations. To make the effect of individual sensors more visible, we had to provide less data points to the agent, i.e. utilize a shorter exploration pattern. To that end we created $C[SA]_{[50]}$ as a subset of $C[SA]$, containing only the first first 50 scheduled controls. To compensate for the presumably less robust output of AgentLoop::Classify, we increased the number of iterations from 5 to 100.

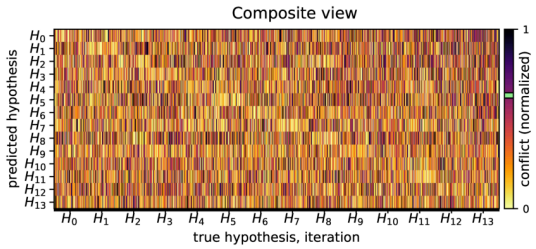
Parameters The modified parameters are given in Table 7.29.

Results We repeated the same experiment but with the modified agent and the modified exploration pattern. The results are depicted in Fig. 7.17 and Table 7.30 with additional figures and tables given in Appendices D.8.7 and D.9.7. Figure 7.17 visualizes the mean intrinsic conflict associated to cases where only a single sensor is enabled. Solely based on visual inspection a substantial difference exists between cases where only the proximity sensor p_P is enabled as given in Figs. 7.17b and 7.17f and cases where only the temperature sensor p_T is enabled as seen in Figs. 7.17d and 7.17h.

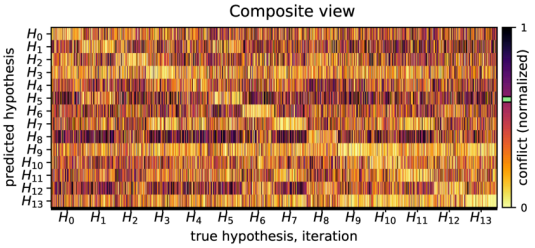
To calculate the impact of a given sensor we divide all results into two sets, one containing all cases where the respective sensor has been activated, the other containing all results where the sensor has been deactivated. Calculating the mean and standard deviation of the classification accuracy of both of these sets allow us to calculate the 'classification gain' i.e. assess how much better the classification accuracy gets when this sensor is added to the agent. The results are displayed in Table 7.31. The temperature sensor p_T stands out as the sensor with the highest impact, followed by the light sensor p_L and the range sensor p_R. For the proximity sensor the



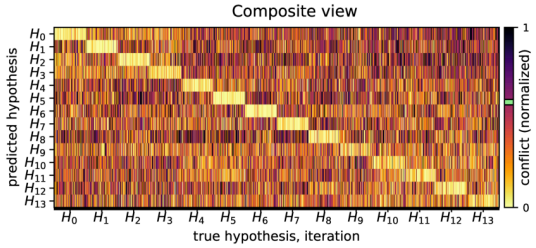
(a) Single sensor $p_L[\bullet]$ on $\mathrm{H}_1^{\oplus}\mathrm{SE}(2)$.



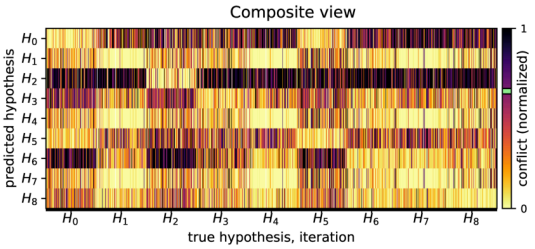
(b) Single sensor $p_P[\bullet]$ on $\mathrm{H}_1^{\oplus}\mathrm{SE}(2)$.



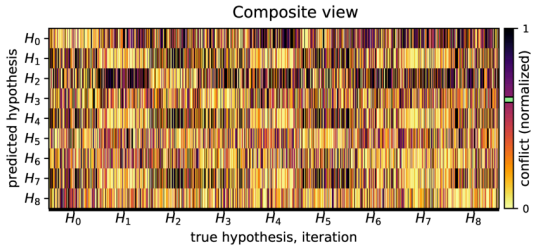
(c) Single sensor $p_R[\bullet]$ on $\mathrm{H}_1^{\oplus}\mathrm{SE}(2)$.



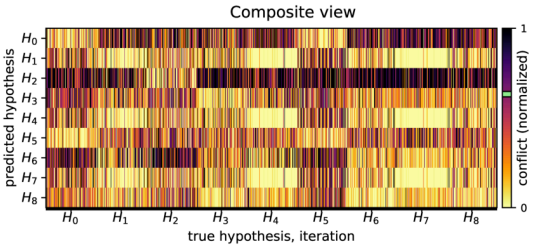
(d) Single sensor $p_T[\bullet]$ on $\mathrm{H}_1^{\oplus}\mathrm{SE}(2)$.



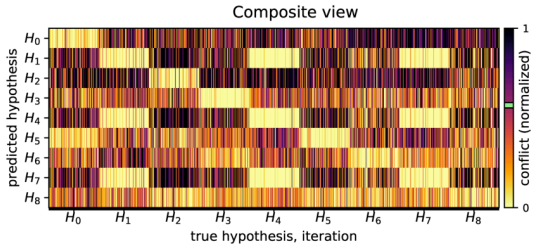
(e) Single sensor $p_L[\bullet]$ on $\mathrm{H}_1^{\ominus}\mathrm{SE}(2)$.



(f) Single sensor $p_P[\bullet]$ on $\mathrm{H}_1^{\ominus}\mathrm{SE}(2)$.



(g) Single sensor $p_R[\bullet]$ on $\mathrm{H}_1^{\ominus}\mathrm{SE}(2)$.



(h) Single sensor $p_T[\bullet]$ on $\mathrm{H}_1^{\ominus}\mathrm{SE}(2)$.

Figure 7.17: Results from Scenario V, sensor permutations without occupancy. Composite result matrices displayed for hypothesis sets $\mathrm{H}_1^{\oplus}\mathrm{SE}(2)$ and $\mathrm{H}_1^{\ominus}\mathrm{SE}(2)$. Individual sensors enabled, respectively. Occupancy data disabled with $p_O[\circ]$.

Sensors				$\Delta[\Xi]/\sum I$		Accuracy
p_L	p_P	p_R	p_T	μ	σ	-
○	○	○	○	nan	nan	0.07
○	○	○	●	0.585	0.299	0.49
○	○	●	○	0.600	0.299	0.27
○	○	●	●	0.574	0.298	0.51
○	●	○	○	0.631	0.310	0.14
○	●	○	●	0.575	0.298	0.45
○	●	●	○	0.593	0.300	0.27
○	●	●	●	0.569	0.298	0.48
●	○	○	○	0.602	0.302	0.32
●	○	○	●	0.571	0.298	0.55
●	○	●	○	0.591	0.301	0.34
●	○	●	●	0.570	0.299	0.54
●	●	○	○	0.588	0.301	0.30
●	●	○	●	0.566	0.298	0.51
●	●	●	○	0.583	0.301	0.33
●	●	●	●	0.565	0.299	0.51

(a) Evaluation on $H_1^{\oplus}SE(2)$.

Sensors				$\Delta[\Xi]/\sum I$		Accuracy
p_L	p_P	p_R	p_T	μ	σ	-
○	○	○	○	nan	nan	0.20
○	○	○	●	0.549	0.384	0.60
○	○	●	○	0.540	0.383	0.38
○	○	●	●	0.548	0.383	0.61
○	●	○	○	0.538	0.390	0.26
○	●	○	●	0.541	0.386	0.56
○	●	●	○	0.534	0.382	0.37
○	●	●	●	0.539	0.383	0.57
●	○	○	○	0.544	0.384	0.45
●	○	○	●	0.544	0.384	0.63
●	○	●	○	0.536	0.384	0.45
●	○	●	●	0.542	0.383	0.61
●	●	○	○	0.539	0.384	0.44
●	●	○	●	0.540	0.384	0.61
●	●	●	○	0.532	0.384	0.46
●	●	●	●	0.537	0.384	0.60

(b) Evaluation on $H_2^{\oplus}SE(2)$.

Sensors				$\Delta[\Xi]/\sum I$		Accuracy
p_L	p_P	p_R	p_T	μ	σ	-
○	○	○	○	nan	nan	0.33
○	○	○	●	0.571	0.364	0.61
○	○	●	○	0.633	0.355	0.44
○	○	●	●	0.578	0.359	0.66
○	●	○	○	0.604	0.365	0.30
○	●	○	●	0.571	0.360	0.59
○	●	●	○	0.613	0.354	0.42
○	●	●	●	0.578	0.357	0.63
●	○	○	○	0.651	0.354	0.52
●	○	○	●	0.590	0.355	0.71
●	○	●	○	0.641	0.353	0.52
●	○	●	●	0.593	0.355	0.69
●	●	○	○	0.621	0.352	0.51
●	●	○	●	0.583	0.353	0.69
●	●	●	○	0.624	0.350	0.51
●	●	●	●	0.589	0.353	0.68

(c) Evaluation on $H_1^{\ominus}SE(2)$.

Sensors				$\Delta[\Xi]/\sum I$		Accuracy
p_L	p_P	p_R	p_T	μ	σ	-
○	○	○	○	nan	nan	0.33
○	○	○	●	0.556	0.344	0.53
○	○	●	○	0.564	0.343	0.35
○	○	●	●	0.552	0.345	0.54
○	●	○	○	0.603	0.353	0.28
○	●	○	●	0.555	0.344	0.51
○	●	●	○	0.563	0.343	0.34
○	●	●	●	0.550	0.346	0.53
●	○	○	○	0.566	0.343	0.38
●	○	○	●	0.546	0.346	0.52
●	○	●	○	0.556	0.343	0.39
●	○	●	●	0.543	0.345	0.53
●	●	○	○	0.563	0.342	0.37
●	●	○	●	0.545	0.346	0.52
●	●	●	○	0.554	0.343	0.38
●	●	●	●	0.544	0.345	0.52

(d) Evaluation on $H_2^{\ominus}SE(2)$.

Table 7.30: Results for Scenario V, sensor permutations without occupancy. Evaluation on hypothesis sets $H_1^{\oplus}SE(2)$, $H_2^{\oplus}SE(2)$, $H_1^{\ominus}SE(2)$ and $H_2^{\ominus}SE(2)$. Occupancy data disabled with $p_O[\circ]$. Intrinsic conflict (μ and σ) and classification accuracy.

Sensor	Accuracy •		Accuracy ∘		Difference • – ∘	
-	μ	σ	μ	σ	μ	σ
p_L	0.425	0.104	0.335	0.160	0.090	0.191
p_P	0.374	0.126	0.386	0.157	-0.013	0.201
p_R	0.406	0.107	0.354	0.166	0.052	0.198
p_T	0.505	0.030	0.255	0.092	0.250	0.096

(a) Evaluation on $\mathrm{H}_1^{\oplus}\mathrm{SE}(2)$.

Sensor	Accuracy •		Accuracy ∘		Difference • – ∘	
-	μ	σ	μ	σ	μ	σ
p_L	0.531	0.082	0.444	0.152	0.088	0.172
p_P	0.484	0.116	0.491	0.141	-0.008	0.183
p_R	0.506	0.096	0.469	0.153	0.037	0.181
p_T	0.599	0.021	0.376	0.091	0.223	0.094

(b) Evaluation on $\mathrm{H}_2^{\oplus}\mathrm{SE}(2)$.

Sensor	Accuracy •		Accuracy ∘		Difference • – ∘	
-	μ	σ	μ	σ	μ	σ
p_L	0.604	0.089	0.497	0.133	0.106	0.160
p_P	0.541	0.126	0.560	0.124	-0.019	0.176
p_R	0.569	0.102	0.532	0.142	0.036	0.175
p_T	0.657	0.040	0.444	0.083	0.214	0.092

(c) Evaluation on $\mathrm{H}_1^{\ominus}\mathrm{SE}(2)$.

Sensor	Accuracy •		Accuracy ∘		Difference • – ∘	
-	μ	σ	μ	σ	μ	σ
p_L	0.451	0.071	0.426	0.103	0.025	0.126
p_P	0.431	0.093	0.446	0.086	-0.015	0.126
p_R	0.448	0.084	0.430	0.094	0.017	0.126
p_T	0.525	0.009	0.353	0.034	0.172	0.035

(d) Evaluation on $\mathrm{H}_2^{\ominus}\mathrm{SE}(2)$.

Table 7.31: Results for Scenario V, sensor permutations without occupancy. Sensor-specific classification impact. Evaluation on hypothesis sets $\mathrm{H}_1^{\oplus}\mathrm{SE}(2)$, $\mathrm{H}_2^{\oplus}\mathrm{SE}(2)$, $\mathrm{H}_1^{\ominus}\mathrm{SE}(2)$ and $\mathrm{H}_2^{\ominus}\mathrm{SE}(2)$. Classification accuracy evaluated per sensor. Classification impact for each sensor calculated as difference between enabled • and disabled ∘ cases. Occupancy data disabled with $p_O[\circ]$.

impact is negative. This means the sensor actually decreases the classification accuracy instead of increasing it, i.e. the agent is better off without it.

7.4.5.8 Summary of Scenario V

Intermediate Conflict Calculation Classification accuracy of 1.0 was given after time step 250 to 500. This corresponds to an exploration pattern with 250 to 500 scheduled controls. Adding additional data points, i.e. continuing to explore the environment, makes the result of AgentLoop::Classify more robust while the impact additional data points have on the mean intrinsic conflict diminishes slightly after time step 800 to 900. To interpret these findings, we have to consider the relation between the agent step size of 1.5 and the size of the environment, given as 20×20×20 (see Section 7.2.2). We assume that a random sequence of length 250 to 500 is likely to have covered the environment considerably well. Thus, exploring the environment any further, only provides information on already explored areas, i.e., data points which have less impact on the overall classification result.

Subset Conflict Calculation Classification accuracy was stable at approximately 1.0 until selecting $n = 8$ subsets. Minimum mean intrinsic conflict was given in most of the cases for $n = 3$ subsets. Minimum T_{CPU}time was given for $n = 2$ or $n = 3$ subsets while minimum T_{WALL} was given for $n = 3$ to $n = 6$ subsets. Considering all results either $n = 2$ subsets or $n = 3$ subsets was identified as a good compromise between accuracy, robustness and computation time.

However this result is implementation dependent, i.e. depend on the way threads are distributed between the individual neighborhoods and instances of the conflict function.

Noise Considering sensor noise, the classification accuracy remained relatively stable up until $\sigma_p = 0.25$ to $\sigma_p = 0.50$. Compared to the results from Scenario IV, AgentLoop::Classify with Δ_{NW} was more susceptible to sensor noise.

Considering actuator noise, the classification accuracy remained stable up until $\sigma_c = 0.05$. Increasing the noise further caused the classification accuracy to drop to approximately chance level accuracy for the individual hypothesis sets at at $\sigma_c = 0.75$ to $\sigma_c = 1.0$. Compared to the results from Scenario IV, the decline in classification accuracy for AgentLoop::Classify using Δ_{NW} starts only at a noise level two to four times as high.

Considering combined sensor and actuator noise, the classification accuracy remains stable up until $\sigma_c, \sigma_p = 0.05$ for almost all cases. Increasing the noise further caused the classification accuracy to drop rapidly while it stayed above chance level accuracy for the individual hypothesis sets in almost all cases. Compared to the results from Scenario IV, the decline in classification accuracy for AgentLoop::Classify using Δ_{NW} starts only at a noise level two to four times as high.

Impact Results showed that occupancy data alone is sufficient for discovering the true hypothesis reasonably well. Disabling occupancy data allowed us to assess the impact of individual sensors on the performance of AgentLoop::Classify. In all cases the temperature sensor p_T stood out as the sensor with the highest impact, followed by the light sensor p_L and the range sensor p_R. The proximity has been identified as a sensor with a negative impact on the classification accuracy. As incorporating its measurements decreases the performance of AgentLoop::Classify instead of increasing it, further usage is discouraged. We will discuss this effect in more detail in the conclusion in Section 8.1. As these results were obtained after having performed Scenario VI, the proximity sensor is part of the sensor set in this and the following scenario.

7.4.6 Scenario VI : Holonomic and Nonholonomic Optimization

The aim of this scenario is to evaluate the performance of the PSO algorithm as given in Section 6.5.2, utilized in the corresponding agent loop as given in Algorithm 16. A notable difference between the classification and the optimization algorithm is the way they handle hypotheses. The classification algorithm utilizes both the matrix Lie group $Q^n_{(\mathcal{H},i)}$ and the corresponding projection matrices $T_{(\cup,i)}$ and $T_{(\cup,i)}$. The optimization algorithm however only utilizes the matrix Lie group $Q^n_{(\mathcal{H},i)}$ and tries to find a projection matrix on its own. Initializing the optimization algorithm with a given hypothesis set therefore ignores the projection matrices.

In the scenarios described in this section we will use both parameterizations discussed in Section 6.2.1 which either utilizes $SO(n)$ (see Eq. (6.6)) or $GL(n, \mathbb{R})$ (see Eq. (6.6)). We will consider both noisy and non noisy scenarios where the noise parameters are selected based on the results from Scenario V (Section 7.4.5).

For all scenarios we initialize the PSO with a fixed set of parameters where most of these parameters have been determined in preliminary experiments and selected as follows:

- The number of particles n_θ, will be set to either 10 or 20.

- The maximal step size $t_{\max}$ is set to 50. As observed in preliminary experiments no substantial improvement of the PSO output after that point.

- The search space boundaries $x_{min}, x_{max} \in \mathbb{R}^z$ are initialized with 3.0 in each dimension. This value was chosen to allow all matrices from $\mathrm{H}_3^{\oplus}\mathrm{SE}(2)$ and $\mathrm{H}_3^{\ominus}\mathrm{SE}(2)$ to be generated as the norm of all corresponding parameter vectors is within this boundary. Thus the search space is a hypercube with an edge length of 6.

- The velocity boundary parameters $v_{min}, v_{max} \in \mathbb{R}^z$ is initialized with 0.3 in each dimension, i.e. five percent of the edge length of the search space hypercube. This value was selected as a compromise between convergence speed and stability.

- The inertia and weigh parameters are set to $w_I = 1.0$, $w_C = 2.0$ and $w_S = 2.0$, based on experiments concluded during the development of the algorithm.

A difficulty in this scenario is that we have to evaluate the optimization approach AgentLoop::Optimize differently than the classification approach AgentLoop::Classify, due to the way these approaches handle hypotheses. The return value of AgentLoop::Classify allows us to assess the performance based on the classification accuracy and assess robustness based on the mean intrinsic conflict. However AgentLoop::Optimize behaves differently. Initialized with the matrix Lie group common to all hypotheses in a given hypothesis set $\mathcal{H}$ (see Algorithm 12), it only yields a single hypothesis, consisting of said matrix Lie group and the projection matrices that minimize the intrinsic conflict in the sensorimotor map (see Algorithm 14). Thus we have neither the classification accuracy nor the mean intrinsic conflict as a foundation for assessing the performance of AgentLoop::Optimize. This means we have to directly compare the true projection matrix with the projection matrix calculated by AgentLoop::Optimize. However, directly comparing these matrices would be the wrong approach, as even matrices that look different may encode the same information. Consider the matrices

$$T_{\cup}^{H_2} = \begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 1.000 & 0.000 \\ 0.000 & 0.000 & 1.000 \end{bmatrix} \quad \text{and} \quad T_{\cup}^{I_1} \begin{bmatrix} 0.555 & 0.832 & -0.002 \\ 0.832 & -0.555 & 0.009 \\ 0.007 & -0.007 & -1.000 \end{bmatrix} \tag{7.7}$$

taken as an example from Table 7.32, This table depicts the results of five iterations for an agent which was initialized with the ground truth hypothesis H_2. It contains the true projection matrix, which for H_2 corresponds to the identity matrix $I_{3\times3}$, as well as the results of the optimization, given as the projection matrices associated to the iterations I_0 to I_4.

Here $T_{\cup}^{H_2}$ denotes the true projection matrix associated to the hypothesis H_2 which is given as $(SE(2), T_{(\cup,2)}, T_{(\cup,2)}^{-1})$ from $\mathrm{H}_3^{\oplus}\mathrm{SE}(2)$ (see Appendix C.3). The matrix $T_{\cup}^{I_1}$ is the projection matrix calculated by AgentLoop::Optimize in iteration I_1, i.e. the output of the PSO for an agent initialized with $SE(2)$ as a hypothesis Lie group and the parameters given in Table 7.33. Though $T_{\cup}^{H_2}$ and $T_{\cup}^{I_1}$ look differently, the sensorimotor maps created based on these projections are similar with respect to the way successive controls interact. To capture this similarity we utilize the Lie bracket as introduced in Section 3.5 (see Eq. (3.58)). We use a similar technique as we used to filter hypotheses in Section 7.2.6. We calculate the commutativity between the vector

Id	Projection Matrix	Commutativity		$\Delta[\Xi_u]$, $\Delta[\circlearrowleft_*]^{\oplus}$	
H_2	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 1.000 & 0.000 \\ 0.000 & 0.000 & 1.000 \end{bmatrix}$	$\circlearrowleft$	[0.00, 1.00, 1.00]		
		$\circlearrowleft_*$	[0.00, 0.71, 0.71]		
I_0	$\begin{bmatrix} 0.555 & 0.832 & -0.002 \\ 0.832 & -0.555 & 0.009 \\ 0.007 & -0.007 & -1.000 \end{bmatrix}$	$\circlearrowleft$	[0.01, 1.00, 1.00]	$\Delta[\Xi_u]$	0.1112
		$\circlearrowleft_*$	[0.01, 0.71, 0.71]	$\Delta[\circlearrowleft_*]^{\oplus}$	0.0044
I_1	$\begin{bmatrix} -0.136 & -0.991 & 0.000 \\ -0.991 & 0.136 & -0.001 \\ 0.000 & -0.000 & -1.000 \end{bmatrix}$	$\circlearrowleft$	[0.00, 1.00, 1.00]	$\Delta[\Xi_u]$	0.1419
		$\circlearrowleft_*$	[0.00, 0.71, 0.71]	$\Delta[\circlearrowleft_*]^{\oplus}$	0.0003
I_2	$\begin{bmatrix} -0.225 & 0.974 & -0.005 \\ 0.974 & 0.225 & 0.000 \\ 0.001 & -0.005 & -1.000 \end{bmatrix}$	$\circlearrowleft$	[0.01, 1.00, 1.00]	$\Delta[\Xi_u]$	0.1316
		$\circlearrowleft_*$	[0.00, 0.71, 0.71]	$\Delta[\circlearrowleft_*]^{\oplus}$	0.0024
I_3	$\begin{bmatrix} -0.957 & -0.290 & 0.002 \\ 0.290 & -0.957 & -0.002 \\ 0.002 & -0.001 & 1.000 \end{bmatrix}$	$\circlearrowleft$	[0.00, 1.00, 1.00]	$\Delta[\Xi_u]$	0.1050
		$\circlearrowleft_*$	[0.00, 0.71, 0.71]	$\Delta[\circlearrowleft_*]^{\oplus}$	0.0012
I_4	$\begin{bmatrix} -0.137 & 0.991 & 0.001 \\ 0.991 & 0.137 & -0.002 \\ -0.002 & 0.001 & -1.000 \end{bmatrix}$	$\circlearrowleft$	[0.00, 1.00, 1.00]	$\Delta[\Xi_u]$	0.1025
		$\circlearrowleft_*$	[0.00, 0.71, 0.71]	$\Delta[\circlearrowleft_*]^{\oplus}$	0.0011

Table 7.32: Example results from Scenario VI, non-noisy optimization parameterized with SO(n). Evaluation on hypothesis set $\mathrm{H}_3^{\oplus}\mathrm{SE}(2)$. True projection matrix and optimization results per iteration I_0 to I_4 for hypothesis H_2 and $n_\theta = 10$ Unnormalized and normalized commutativity vector given as $\circlearrowleft$ and $\circlearrowleft_*$. Unnormalized intrinsic conflict denoted $\Delta[\Xi_u]$. Commutativity delta denoted $\Delta[\circlearrowleft_*]$.

fields, associated to the agent's unit control vectors. Consider $\hat{c}_1, \hat{c}_2, \hat{c}_3 \in C_{\circledcirc}$ with $\hat{c}_1 = [1,0,0]^T$, $\hat{c}_2 = [0,1,0]^T$ and $\hat{c}_3 = [0,0,1]^T$. The hypothesis matrix Lie group is $SE(2)$ with the generators $\hat{\mathfrak{g}}_1, \hat{\mathfrak{g}}_2, \hat{\mathfrak{g}}_3$ given in Appendix A.2.1. Considering $T^{H_2}_{\circlearrowleft}$, the control vector $\hat{c}_1$ is mapped to the generator $\hat{\mathfrak{g}}_1$, $\hat{c}_2$ is mapped to $\hat{\mathfrak{g}}_2$ and $\hat{c}_3$ is mapped to $\hat{\mathfrak{g}}_3$. Calculating the pairwise degree of commutativity of $\hat{c}_1, \hat{c}_2, \hat{c}_3$ using the Lie bracket yields

$$
\begin{aligned}
\|[\hat{c}_1, \hat{c}_2]\| &= \|[\hat{\mathfrak{g}}_1, \hat{\mathfrak{g}}_2]\| = \|0_{3\times 3}\| = 0.00 \\
\|[\hat{c}_1, \hat{c}_3]\| &= \|[\hat{\mathfrak{g}}_1, \hat{\mathfrak{g}}_3]\| = \| - \hat{\mathfrak{g}}_2\| = 1.00 \\
\|[\hat{c}_2, \hat{c}_3]\| &= \|[\hat{\mathfrak{g}}_2, \hat{\mathfrak{g}}_3]\| = \| - \hat{\mathfrak{g}}_1\| = 1.00
\end{aligned}
$$

where $\|\cdot\|$ denotes the Frobenius norm [Lee, 2003, p. 638] as the square root of the sum of all squared matrix entries. The commutativity vector $\circlearrowleft^{H_2}$ associated to the true projection matrix $T^{H_2}_{\circlearrowleft}$ is given as

$$
\circlearrowleft^{H_2} = [\, \|[\hat{\mathfrak{g}}_1, \hat{\mathfrak{g}}_2]\|, \|[\hat{\mathfrak{g}}_1, \hat{\mathfrak{g}}_3]\|, \|[\hat{\mathfrak{g}}_2, \hat{\mathfrak{g}}_3]\| \,] = [\, 0.00,\ 1.00,\ 1.00 \,]
$$

and the normalized commutativity vector $\circlearrowleft^T_*$ as

$$
\circlearrowleft^{H_2}_* = [\, 0.00,\ 0.71,\ 0.71 \,].
$$

We now consider $T^{I_1}_{\circlearrowleft}$ where each control vector is mapped to a different linear combination of the three generators $\hat{\mathfrak{g}}_1$, $\hat{\mathfrak{g}}_2$ and $\hat{\mathfrak{g}}_3$. Notably $\hat{c}_1$ is not only mapped to $\hat{\mathfrak{g}}_1$ but to $\hat{\mathfrak{g}}_2$ and $\hat{\mathfrak{g}}_3$ as well. The same holds for $\hat{c}_2$ which is mapped not only to $\hat{\mathfrak{g}}_2$ but also to $\hat{\mathfrak{g}}_1$ and $\hat{\mathfrak{g}}_3$. These linear combinations of $\hat{\mathfrak{g}}_1$, $\hat{\mathfrak{g}}_2$ and $\hat{\mathfrak{g}}_3$ correspond to vector fields on $SE(2)$, given as elements of its Lie algebra (see Sections 3.4 to 3.6). The vector fields associated to the controls are

$$
\begin{aligned}
\hat{c}_1 : \mathcal{X}_1 &= 0.555\,\hat{\mathfrak{g}}_1 + 0.832\,\hat{\mathfrak{g}}_2 + 0.007\,\hat{\mathfrak{g}}_3 \\
\hat{c}_2 : \mathcal{X}_2 &= 0.832\,\hat{\mathfrak{g}}_1 - 0.555\,\hat{\mathfrak{g}}_2 - 0.007\,\hat{\mathfrak{g}}_3 \\
\hat{c}_3 : \mathcal{X}_3 &= -0.002\,\hat{\mathfrak{g}}_1 + 0.009\,\hat{\mathfrak{g}}_2 - 1.000\,\hat{\mathfrak{g}}_3
\end{aligned}
$$

Calculating the pairwise degree of commutativity of $\hat{c}_1, \hat{c}_2, \hat{c}_3$ using the Lie bracket yields

$$\begin{aligned} |||[\hat{c}_1, \hat{c}_2]||| &= |||[\mathcal{X}_1, \mathcal{X}_2]||| = 0.01 \\ |||[\hat{c}_1, \hat{c}_3]||| &= |||[\mathcal{X}_1, \mathcal{X}_3]||| = 1.00 \\ |||[\hat{c}_2, \hat{c}_3]||| &= |||[\mathcal{X}_3, \mathcal{X}_3]||| = 1.00. \end{aligned}$$

The commutativity vector $\circlearrowleft^{I_1}$ associated to the hypothesis projection matrix $T^{I_1}_{\circlearrowleft}$ is given as

$$\circlearrowleft^{I_1} = [\, ||\, [\hat{\mathfrak{g}}_1, \hat{\mathfrak{g}}_2]\, ||, ||\, [\hat{\mathfrak{g}}_1, \hat{\mathfrak{g}}_3]\, ||, ||\, [\hat{\mathfrak{g}}_2, \hat{\mathfrak{g}}_3]\, || \,] = [\, 0.01,\ 1.00,\ 1.00\,]$$

and the normalized commutativity vector $\circlearrowleft^{I}_{*}$ as

$$\circlearrowleft^{I_1}_{*} = [\, 0.01,\ 0.71,\ 0.71\,].$$

These commutativity vectors indicate how much individual controls commute. The degree to which the vector fields associated to the agent's controls commute, directly affects the structure of the sensorimotor map. This is due to the fact that the way a sensorimotor chain Ω_t is projected into a sensorimotor neighborhood, depends on the effect successive group operations have. This effect is reflected in the commutativity vector which in this way provides a 'commutativity signature' of a certain hypothesis. This means that hypotheses with the same or a similar commutativity vector construct sensorimotor maps that are similar with respect to the way how projections are calculated and ultimately shortcuts are established as discussed in the example in Section 5.3.1. To calculate a measure of similarity between $\circlearrowleft^{H_2}_{*}$ and $\circlearrowleft^{I_1}_{*}$ we utilize the dot product. As both vectors are normalized, the dot product returns the cosine of the angle between them. This means that if $\circlearrowleft^{H_2}_{*}$ and $\circlearrowleft^{I_1}_{*}$ are similar, the dot product will be close to zero. However, the more these vectors differ, the more will the dot product approach one. To make this nonlinear relation between vector alignment and dot product linear, we calculate the angle between these vectors by applying arccos and normalizing the return value to the range $[0, 1]$. The commutativity delta is then given as

$$\Delta[\circlearrowleft_{*}]^{\oplus}(\circlearrowleft^{i}_{*}, \circlearrowleft^{j}_{*}) = \frac{2}{\pi} |\, \arccos(\circlearrowleft^{i}_{*} \cdot \circlearrowleft^{j}_{*})\, |,$$

which in this case yields 0.0044.

Considering the results of all evaluations with a holonomic setup (see Appendices D.10 and D.10.3), we utilize the commutativity delta to assign a measure of 'quality' to the outcome of the optimization. With respect to the degree of similarity between the original and the determined matrices we see that a commutativity delta of $\Delta[\circlearrowleft_{*}]^{\oplus} \in [0.0, 0.01]$ corresponds to matrices with a high similarity to the true matrix. For a commutativity delta of $\Delta[\circlearrowleft_{*}]^{\oplus} \in (0.01, 0.03]$ similarity between the determined matrices and the true matrix is still given in most cases. However, for a commutativity delta of $\Delta[\circlearrowleft_{*}]^{\oplus} \in (0.03, 1.00]$ we see that the determined matrices considerably differ from the true matrix.

For the nonholonomic scenarios the calculation is slightly different. In this case the configuration space of the agent is given as $C \equiv \mathbb{R}^2$, and thus the agent's control vector has only two entries. Therefore, the commutativity vector $\circlearrowleft$ has only a single entry corresponding to

num_iterations	5
dim_controls	3 and 2
noise_actuator	0.0
enable_sensors	true
noise_sensors	0.0
agent_stepsize	1.5 (Scenario II ,Section 7.4.2)
exploration_pattern	$C[SA]$ (Scenario II ,Section 7.4.2)
conflict_function	Δ_{NW}
conflict_sigma_factor	0.3 (Scenario I ,Section 7.4.1)
conflict_subsets	(1, 1.0)
bootstrapping_type	optimize
pso_num_particles	10 and 20
pso_parameterization	$\Upsilon[\mathcal{Q}^n_{\mathcal{H}}]^{\ominus}$
hypothesis_group	$SE(2)$
hypothesis_set	$\text{H}^{\oplus}_3\text{SE}(2)$ and $\text{H}^{\ominus}_3\text{SE}(2)$

Table 7.33: Parameters for Scenario VI, non-noisy optimization parameterized with SO(n).

$\| [\hat{c}_1, \hat{c}_2] \|$. As a consequence, we can neither normalize this vector, nor calculate the dot product. In these cases the commutativity delta is calculated as

$$\Delta[\circledcirc_*]^{\ominus}(\circledcirc^i, \circledcirc^j) = \mid \circledcirc^i - \circledcirc^j \mid.$$

Considering the results of all evaluations with a nonholonomic setup (see Appendices D.11 and D.11.3), we utilize the commutativity delta to assign a measure of 'quality' to the outcome of the optimization. With respect to the degree of similarity between the original and the determined matrices we see that a commutativity delta of $\Delta[\circledcirc_*]^{\ominus} \in [0.0, 0.05]$ corresponds to matrices with a high similarity to the true matrix. For a commutativity delta of $\Delta[\circledcirc_*]^{\ominus} \in (0.05, 0.06]$ similarity between the determined matrices and the true matrix is lowered. However, for a commutativity delta of $\Delta[\circledcirc_*]^{\ominus} > 0.6$ we see that the determined matrices considerably differ from the true matrix.

We utilize $\Delta[\circledcirc_*]^{\oplus}$ and $\Delta[\circledcirc_*]^{\ominus}$ to evaluate the results of AgentLoop::Optimize. In Section 7.4.6.1 and Section 7.4.6.2 we evaluate optimization AgentLoop::Optimize in a non-noisy setting using the two different parameterizations of the projection matrices $\Upsilon[\mathcal{Q}^n_{\mathcal{H}}]^{\ominus}$ and $\Upsilon[\mathcal{Q}^n_{\mathcal{H}}]^{\oplus}$ respectively. In Section 7.4.6.3 we evaluate AgentLoop::Optimize in conjunction with $\Upsilon[\mathcal{Q}^n_{\mathcal{H}}]^{\ominus}$ in a noisy setting.

7.4.6.1 Non-Noisy Optimization parameterized with SO(n)

Parameters We initialize the scenario with the parameters given in Table 7.33.

Results Exemplary results are displayed in Tables 7.34 and 7.35 where additionally tables for both the holonomic and nonholonmic case are given in Appendices D.10.1 and D.11.1 respectively. We first consider the holonomic case displayed in Table 7.34. We see that for $n_\theta = 10$ (Table 7.34a) in most cases $\Delta[\circledcirc_*]^{\oplus}$ is very close to zero with its mean and standard deviation over all hypotheses and iterations being $(0.0088, 0.0215)$ respectively. The exceptions are (H_0, I_0) with $\Delta[\circledcirc_*]^{\oplus} = 0.0285$ and (H_1, I_1) with $\Delta[\circledcirc_*]^{\oplus} = 0.0824$. Increasing the number of particles, as displayed in Table 7.34b, improves the performance of AgentLoop::Optimize in this

Id	Hypothesis, $\Delta[\circledcirc_*]^{\oplus}$		
	H_0	H_1	H_2
I_0	0.0285	0.0014	0.0044
I_1	0.0012	0.0824	0.0003
I_2	0.0004	0.0017	0.0024
I_3	0.0034	0.0002	0.0012
I_4	0.0014	0.0025	0.0011
μ	0.0070	0.0176	0.0019
σ	0.0108	0.0324	0.0014
$\mu = 0.0088, \sigma = 0.0215$			

(a) Evaluation for $n_\theta = 10$.

Id	Hypothesis, $\Delta[\circledcirc_*]^{\oplus}$		
	H_0	H_1	H_2
I_0	0.0023	0.0003	0.0007
I_1	0.0001	0.0029	0.0027
I_2	0.0001	0.0003	0.0009
I_3	0.0003	0.0020	0.0019
I_4	0.0037	0.0034	0.0004
μ	0.0013	0.0018	0.0013
σ	0.0015	0.0013	0.0009
$\mu = 0.0015, \sigma = 0.0013$			

(b) Evaluation for $n_\theta = 20$.

Table 7.34: Results from Scenario VI, non-noisy optimization parameterized with SO(n) on hypothesis set $\mathrm{H}_3^{\oplus}\mathrm{SE}(2)$. Commutativity delta $\Delta[\circledcirc_*]^{\oplus}$ per iteration I_0 to I_4. Mean and standard deviation of $\Delta[\circledcirc_*]^{\oplus}$ per hypothesis H_0 to H_2 and over all hypotheses.

Id	Hypothesis, $\Delta[\circledcirc_*]^{\ominus}$				
	H_0	H_1	H_2	H_3	H_4
I_0	0.0092	0.0002	0.0025	0.0112	0.0010
I_1	0.0857	0.0019	0.0213	0.0054	0.0000
I_2	0.0086	0.0051	0.0052	0.0057	0.0007
I_3	0.0025	0.0003	0.0037	0.3537	0.0520
I_4	0.0069	0.0022	0.0020	0.0068	0.0003
μ	0.0226	0.0019	0.0069	0.0766	0.0108
σ	0.0316	0.0018	0.0073	0.1386	0.0206
$\mu = 0.0238, \sigma = 0.0713$					

(a) Evaluation for $n_\theta = 10$.

Id	Hypothesis, $\Delta[\circledcirc_*]^{\ominus}$				
	H_0	H_1	H_2	H_3	H_4
I_0	0.0033	0.0042	0.0019	0.0043	0.0003
I_1	0.0037	0.0046	0.0014	0.2906	0.0005
I_2	0.0027	0.0003	0.0097	0.1733	0.0008
I_3	0.0108	0.0025	0.0057	0.0028	0.0000
I_4	0.0045	0.0004	0.0044	0.2700	0.0000
μ	0.0050	0.0024	0.0046	0.1482	0.0003
σ	0.0030	0.0018	0.0030	0.1246	0.0003
$\mu = 0.0321, \sigma = 0.0822$					

(b) Evaluation for $n_\theta = 20$.

Table 7.35: Results from Scenario VI, non-noisy optimization parameterized with SO(n) on hypothesis set $\mathrm{H}_3^{\ominus}\mathrm{SE}(2)$. Commutativity delta $\Delta[\circledcirc_*]^{\ominus}$ per iteration I_0 to I_4. Mean and standard deviation of $\Delta[\circledcirc_*]^{\ominus}$ per hypothesis H_0 to H_4 and over all hypotheses.

case as mean and standard deviation of $\Delta[\circledcirc_*]^{\oplus}$ amount to $(0.0015, 0.0013)$, respectively. We see that in most cases the result of AgentLoop::Optimize is close to the true projection matrix (see also Appendix D.10.1).

The nonholonomic case is displayed in Table 7.35. For $n_\theta = 10$ (Table 7.35a) mean and standard deviation over all hypotheses and iterations for $\Delta[\circledcirc_*]^{\ominus}$ are given as $(0.0238, 0.0713)$ respectively. However, in most cases $\Delta[\circledcirc_*]^{\ominus}$ is considerably lower with (H_0, I_0), (H_2, I_1), (H_4, I_2) being small and (H_3, I_3) being an exception. Increasing the number of particles, displayed in Table 7.35b lowers the overall performance of AgentLoop::Optimize in this case, as both mean and standard deviation of $\Delta[\circledcirc_*]^{\ominus}$ increase to $(0.0321, 0.0822)$ respectively. However, comparing $n_\theta = 10$ with $n_\theta = 20$ for individual hypotheses, we see that for H_0, H_1, H_2 and H_4 increasing the number of particles, improves the performance of AgentLoop::Optimize. We see that in most cases the result of AgentLoop::Optimize is close to the true projection matrix (see also Appendix D.11.1).

An exception is the identity projection matrix H_3, which in both $n_\theta = 10$ with $n_\theta = 20$ is difficult to detect. For $n_\theta = 20$ the cases (H_3, I_1), (H_3, I_2) and (H_3, I_4) are considerably worse than in $n_\theta = 10$ which explains the difference in overall performance of AgentLoop::Optimize in Table 7.35b. However, based on our results we can not say for sure whether the poor perfor-

num_iterations	5
dim_controls	3 and 2
noise_actuator	0.0
enable_sensors	true
noise_sensors	0.0
agent_stepsize	1.5 (Scenario II ,Section 7.4.2)
exploration_pattern	$C[SA]$ (Scenario II ,Section 7.4.2)
conflict_function	Δ_{NW}
conflict_sigma_factor	0.3 (Scenario I ,Section 7.4.1)
conflict_subsets	(1, 1.0)
bootstrapping_type	optimize
pso_num_particles	10 and 20
pso_parameterization	$\Upsilon[Q^n_{\mathcal{H}}]^{\oplus}$
hypothesis_group	$SE(2)$
hypothesis_set	$\text{H}^{\oplus}_3\text{SE}(2)$ and $\text{H}^{\ominus}_3\text{SE}(2)$

Table 7.36: Parameters for Scenario VI, non-noisy optimization parameterized with GL(n,R).

Id	Hypothesis, $\Delta[\textcircled{D}_*]^{\oplus}$		
	H_0	H_1	H_2
I_0	0.4638	0.4208	0.2672
I_1	0.0747	0.2112	0.3813
I_2	0.3940	0.2728	0.5705
I_3	0.1089	0.3238	0.1338
I_4	0.2322	0.0864	0.1051
μ	0.2547	0.2630	0.2916
σ	0.1532	0.1119	0.1709
$\mu = 0.2698, \sigma = 0.1535$			

(a) Evaluation for $n_\theta = 10$.

Id	Hypothesis, $\Delta[\textcircled{D}_*]^{\oplus}$		
	H_0	H_1	H_2
I_0	0.4921	0.5643	0.2797
I_1	0.1843	0.0803	0.6129
I_2	0.1223	0.1110	0.2578
I_3	0.0236	0.4197	0.1186
I_4	0.0395	0.1387	0.1195
μ	0.1724	0.2628	0.2777
σ	0.1701	0.1935	0.1806
$\mu = 0.2376, \sigma = 0.1941$			

(b) Evaluation for $n_\theta = 20$.

Table 7.37: Results from Scenario VI, non-noisy optimization parameterized with GL(n,R) on hypothesis set $\text{H}^{\oplus}_3\text{SE}(2)$. Commutativity delta $\Delta[\textcircled{D}_*]^{\oplus}$ per iteration I_0 to I_4. Mean and standard deviation of $\Delta[\textcircled{D}_*]^{\oplus}$ per hypothesis H_0 to H_2 and over all hypotheses.

mance of AgentLoop::Optimize in this case is related to the number of particles, the projection matrix or could considered to be an outlier.

7.4.6.2 Non-Noisy Optimization parameterized with GL(n,R)

Parameters We initialize the scenario with the parameters given in Table 7.36.

Results Exemplary results are displayed in Tables 7.37 and 7.38 where additionally tables for both the holonomic and nonholonmic case are given in Appendices D.10.2 and D.11.2 respectively. We first consider the holonomic case displayed in Table 7.37. Comparing both $n_\theta = 10$ (Table 7.37a) and $n_\theta = 20$ (Table 7.37b) with the results from the previous scenario, we see that the performance of AgentLoop::Optimize is considerably worse when utilizing $GL(n, \mathbb{R})$. Mean and standard deviation of $\Delta[\textcircled{D}_*]^{\oplus}$ over all hypotheses is $(0.2698, 0.1535)$ for $n_\theta = 10$ and is $(0.2376, 0.1941)$ for $n_\theta = 20$. We see that in most cases the result of AgentLoop::Optimize considerably differ from the true projection matrix (see also Appendix D.10.2).

Id	Hypothesis, $\Delta[\circledast_*]^{\ominus}$				
	H_0	H_1	H_2	H_3	H_4
I_0	0.8855	2.1988	1.0216	1.3606	0.1664
I_1	6.4784	2.9323	4.4226	6.8813	5.0573
I_2	2.2289	0.0818	0.3695	4.1776	1.0209
I_3	0.3132	0.3825	0.2347	0.0001	0.5073
I_4	0.0473	0.1816	0.5097	1.0967	0.1802
μ	1.9907	1.1554	1.3116	2.7033	1.3864
σ	2.3668	1.1785	1.5781	2.5028	1.8615
$\mu = 1.7095, \sigma = 2.0840$					

(a) Evaluation for $n_\theta = 10$.

Id	Hypothesis, $\Delta[\circledast_*]^{\ominus}$				
	H_0	H_1	H_2	H_3	H_4
I_0	1.9190	0.1648	0.9120	0.0036	1.4033
I_1	0.0162	0.1787	0.0333	0.7064	0.6852
I_2	0.4228	0.5077	0.0562	0.1662	0.5170
I_3	0.2075	0.2013	0.2668	0.0001	3.1062
I_4	0.0873	0.0029	0.6002	0.0028	0.4619
μ	0.5306	0.2111	0.3737	0.1758	1.2347
σ	0.7078	0.1641	0.3374	0.2728	0.9945
$\mu = 0.5052, \sigma = 0.7145$					

(b) Evaluation for $n_\theta = 20$.

Table 7.38: Results from Scenario VI, non-noisy optimization parameterized with GL(n,R) on hypothesis set $\mathrm{H}_3^{\ominus}\mathrm{SE}(2)$. Commutativity delta $\Delta[\circledast_*]^{\ominus}$ per iteration I_0 to I_4. Mean and standard deviation of $\Delta[\circledast_*]^{\ominus}$ per hypothesis H_0 to H_4 and over all hypotheses.

`num_iterations`	5
`dim_controls`	3 and 2
`noise_actuator`	0.05, 0.1, 0.25 and 0.5 (Scenario V ,Section 7.4.5)
`enable_sensors`	`true`
`noise_sensors`	sensor noise (all sensors) matches actuator noise
`agent_stepsize`	1.5 (Scenario II ,Section 7.4.2)
`exploration_pattern`	$C[SA]$ (Scenario II ,Section 7.4.2)
`conflict_function`	Δ_{NW}
`conflict_sigma_factor`	0.3 (Scenario I ,Section 7.4.1)
`conflict_subsets`	(1, 1.0)
`bootstrapping_type`	`optimize`
`pso_num_particles`	10 and 20
`pso_parameterization`	$\Upsilon[Q^n_{\mathcal{H}}]^{\ominus}$
`hypothesis_group`	$SE(2)$
`hypothesis_set`	$\mathrm{H}_3^{\oplus}\mathrm{SE}(2)$ and $\mathrm{H}_3^{\ominus}\mathrm{SE}(2)$

Table 7.39: Parameters for Scenario VI, noisy optimization parameterized with SO(n).

The nonholonomic case is displayed in Table 7.38. However, the results are comparable to the results from the holonomic case. Comparing both $n_\theta = 10$ (Table 7.38a) and $n_\theta = 20$ (Table 7.38b) with the results from the previous scenario, we see that the performance of AgentLoop::Optimize is considerably worse in this setting as well.

7.4.6.3 Noisy Optimization parameterized with SO(n)

In a final scenario we evaluate AgentLoop::Optimize in a setting with noisy actuators and sensors. As the performance of AgentLoop::Optimize parameterized with $GL(n, \mathbb{R})$ was subpar, we only consider the parameterization based on $SO(n)$. We utilize noise in the range of $\sigma_c, \sigma_p = 0.05$ to $\sigma_c, \sigma_p = 0.50$. This was the range we identified in Scenario V as a turning point between situations in which the classifier could identify the true hypothesis and situations in which it could not any more.

Parameters We initialize the scenario with the parameters given in Table 7.39.

Results Exemplary results are displayed in Tables 7.40 and 7.41 where additionally tables for both the holonomic and nonholonmic case are given in Appendices D.10.3 and D.11.3, respectively.

We first consider the holonomic case displayed in Table 7.40. For $\sigma_c, \sigma_p = 0.05$ mean and standard deviation of $\Delta[\circledcirc_*]^{\oplus}$ are given as $(0.0108, 0.0083)$ and $(0.0137, 0.0130)$ respectively. While AgentLoop::Optimize shows a worse performance than in the non-noisy case, in most cases the result of AgentLoop::Optimize is close to the true projection matrix (see also Appendix D.10.3). As expected, with increasing noise, the performance of AgentLoop::Optimize gets worse. For $\sigma_c, \sigma_p = 0.10$ mean and standard deviation of $\Delta[\circledcirc_*]^{\oplus}$ are $(0.0364, 0.0458)$ and $(0.0292, 0.0371)$, respectively, while similarity between identified projection matrices and the true projection matrix is still given in most cases (see also Appendix D.10.3). At the higher noise levels $\sigma_c, \sigma_p = 0.25$ and $\sigma_c, \sigma_p = 0.50$, however, AgentLoop::Optimize fails to identify valid projection matrices. For $\sigma_c, \sigma_p = 0.25$, mean and standard deviation of $\Delta[\circledcirc_*]^{\oplus}$ are given as $(0.0972, 0.0983)$ and $(0.0539, 0.0296)$ respectively. For $\sigma_c, \sigma_p = 0.50$, mean and standard deviation of $\Delta[\circledcirc_*]^{\oplus}$ are given as $(0.2077, 0.1732)$ and $(0.2365, 0.1717)$ respectively. In both cases the identified projection matrices displayed considerably differ from the true projection matrix (see also Appendix D.10.3).

The nonholonomic case is displayed in Table 7.41. For $\sigma_c, \sigma_p = 0.05$ mean and standard deviation of $\Delta[\circledcirc_*]^{\ominus}$ are given as $(0.0569, 0.1062)$ and $(0.0100, 0.0084)$ respectively. The results are comparable to the noholonomic case. We see, that in most cases the result of AgentLoop::Optimize is close to the true projection matrix (see also Appendix D.11.3).

For $\sigma_c, \sigma_p = 0.10$ this still holds as mean and standard deviation of $\Delta[\circledcirc_*]^{\ominus}$ are given as $(0.0427, 0.0563)$ and $(0.0595, 0.0924)$ respectively. Similarity between identified projection matrices and the true projection matrix is still given in most cases (see also Appendix D.10.3).

Similar as in the holonomic case at noise levels of $\sigma_c, \sigma_p = 0.25$ and $\sigma_c, \sigma_p = 0.50$, however, AgentLoop::Optimize fails to identify valid projection matrices. For $\sigma_c, \sigma_p = 0.25$, mean and standard deviation of $\Delta[\circledcirc_*]^{\ominus}$ are given as $(0.1171, 0.0982)$ and $(0.0811, 0.0624)$ respectively. For $\sigma_c, \sigma_p = 0.50$, mean and standard deviation of $\Delta[\circledcirc_*]^{\ominus}$ are given as $(0.2613, 0.2363)$ and $(0.3393, 0.2784)$ respectively. In both cases the identified projection matrices considerably differ from the true projection matrix (see also Appendix D.10.3).

7.4.6.4 Summary of Scenario VI

Considering the results depicted in Section 7.4.6.1 and Section 7.4.6.2, we see that the performance of AgentLoop::Optimize considerably depends on the selected search space. In scenarios where $SO(n)$ was utilized, AgentLoop::Optimize was able to identify the true projection matrix by utilizing the sensorimotor map$\Lambda[\Xi_u]$ as an objective function. However, the performance on $GL(n, \mathbb{R})$ was poor as AgentLoop::Optimize failed to find a suitable parameterization in most cases. As cases exist in which the algorithm found such a matrix in the $GL(n, \mathbb{R})$ case, the question arises how to improve the performance of AgentLoop::Optimize for these scenarios.

Considering the results depicted in Section 7.4.6.3, we see that AgentLoop::Optimize can to some extent be utilized in noisy scenarios. Up to a combined noise level for the actuator and the sensor of $\sigma_c, \sigma_p = 0.10$, AgentLoop::Optimize was able to find projection matrices similar to

Id	Hypothesis, $\Delta[\circledcirc_*]^{\oplus}$		
	H_0	H_1	H_2
I_0	0.0108	0.0035	0.0054
I_1	0.0004	0.0215	0.0315
I_2	0.0079	0.0056	0.0196
I_3	0.0079	0.0105	0.0066
I_4	0.0167	0.0038	0.0108
μ	0.0087	0.0090	0.0148
σ	0.0053	0.0067	0.0097
$\mu = 0.0108, \sigma = 0.0083$			

(a) Evaluation for $\sigma_c, \sigma_p = 0.05$ and $n_\theta = 10$.

Id	Hypothesis, $\Delta[\circledcirc_*]^{\oplus}$		
	H_0	H_1	H_2
I_0	0.0002	0.0001	0.0052
I_1	0.0033	0.0243	0.0410
I_2	0.0099	0.0052	0.0173
I_3	0.0055	0.0157	0.0094
I_4	0.0218	0.0066	0.0396
μ	0.0081	0.0104	0.0225
σ	0.0075	0.0086	0.0151
$\mu = 0.0137, \sigma = 0.0130$			

(b) Evaluation for $\sigma_c, \sigma_p = 0.05$ and $n_\theta = 20$.

Id	Hypothesis, $\Delta[\circledcirc_*]^{\oplus}$		
	H_0	H_1	H_2
I_0	0.0251	0.0176	0.0131
I_1	0.0143	0.0269	0.1072
I_2	0.0036	0.0126	0.0222
I_3	0.0297	0.1714	0.0298
I_4	0.0024	0.0083	0.0615
μ	0.0150	0.0474	0.0468
σ	0.0110	0.0623	0.0343
$\mu = 0.0364, \sigma = 0.0458$			

(c) Evaluation for $\sigma_c, \sigma_p = 0.10$ and $n_\theta = 10$.

Id	Hypothesis, $\Delta[\circledcirc_*]^{\oplus}$		
	H_0	H_1	H_2
I_0	0.0261	0.0139	0.0118
I_1	0.0136	0.0107	0.0273
I_2	0.0039	0.0034	0.0237
I_3	0.0257	0.1297	0.0287
I_4	0.0051	0.0091	0.1050
μ	0.0149	0.0334	0.0393
σ	0.0096	0.0483	0.0334
$\mu = 0.0292, \sigma = 0.0371$			

(d) Evaluation for $\sigma_c, \sigma_p = 0.10$ and $n_\theta = 20$.

Id	Hypothesis, $\Delta[\circledcirc_*]^{\oplus}$		
	H_0	H_1	H_2
I_0	0.1291	0.1025	0.0745
I_1	0.0561	0.0949	0.1562
I_2	0.1157	0.0241	0.3985
I_3	0.0294	0.1022	0.0209
I_4	0.0041	0.0758	0.0744
μ	0.0669	0.0799	0.1449
σ	0.0484	0.0295	0.1340
$\mu = 0.0972, \sigma = 0.0938$			

(e) Evaluation for $\sigma_c, \sigma_p = 0.25$ and $n_\theta = 10$.

Id	Hypothesis, $\Delta[\circledcirc_*]^{\oplus}$		
	H_0	H_1	H_2
I_0	0.0671	0.0095	0.1004
I_1	0.0754	0.0375	0.0599
I_2	0.0141	0.0323	0.0750
I_3	0.0363	0.0947	0.0217
I_4	0.0924	0.0527	0.0389
μ	0.0571	0.0453	0.0592
σ	0.0282	0.0283	0.0274
$\mu = 0.0539, \sigma = 0.0296$			

(f) Evaluation for $\sigma_c, \sigma_p = 0.25$ and $n_\theta = 20$.

Id	Hypothesis, $\Delta[\circledcirc_*]^{\oplus}$		
	H_0	H_1	H_2
I_0	0.1006	0.0949	0.5519
I_1	0.0677	0.0085	0.4961
I_2	0.2029	0.2928	0.1985
I_3	0.0085	0.1689	0.1384
I_4	0.2080	0.1045	0.4740
μ	0.1175	0.1339	0.3718
σ	0.0776	0.0944	0.1690
$\mu = 0.2077, \sigma = 0.1732$			

(g) Evaluation for $\sigma_c, \sigma_p = 0.50$ and $n_\theta = 10$.

Id	Hypothesis, $\Delta[\circledcirc_*]^{\oplus}$		
	H_0	H_1	H_2
I_0	0.2154	0.0840	0.3652
I_1	0.3298	0.1477	0.4966
I_2	0.0616	0.2917	0.1384
I_3	0.0004	0.1686	0.5828
I_4	0.2352	0.0367	0.3937
μ	0.1685	0.1457	0.3953
σ	0.1203	0.0866	0.1499
$\mu = 0.2365, \sigma = 0.1717$			

(h) Evaluation for $\sigma_c, \sigma_p = 0.50$ and $n_\theta = 20$.

Table 7.40: Results from Scenario VI, noisy optimization parameterized with SO(n) on hypothesis set $\mathrm{H}_3^{\oplus}\mathrm{SE}(2)$. Sensor and actuator noise ranging from $\sigma_c, \sigma_p = 0.05$ to $\sigma_c, \sigma_p = 0.50$ Commutativity delta $\Delta[\circledcirc_*]^{\oplus}$ per iteration I_0 to I_4. Mean and standard deviation of $\Delta[\circledcirc_*]^{\oplus}$ per hypothesis H_0 to H_2 and over all hypotheses.

Id	Hypothesis, $\Delta[\circledast_*]^{\ominus}$				
	H_0	H_1	H_2	H_3	H_4
I_0	0.0039	0.0311	0.0054	0.0113	0.0011
I_1	0.0019	0.2978	0.0010	0.2982	0.0016
I_2	0.0024	0.0360	0.0118	0.0164	0.0004
I_3	0.0254	0.3014	0.0030	0.2774	0.0002
I_4	0.0021	0.0198	0.0407	0.0131	0.0202
μ	0.0071	0.1372	0.0124	0.1233	0.0047
σ	0.0092	0.1327	0.0146	0.1345	0.0078
$\mu = 0.0569, \sigma = 0.1062$					

(a) Evaluation for $\sigma_c, \sigma_p = 0.05$ and $n_\theta = 10$.

Id	Hypothesis, $\Delta[\circledast_*]^{\ominus}$				
	H_0	H_1	H_2	H_3	H_4
I_0	0.0107	0.0152	0.0056	0.0243	0.0026
I_1	0.0035	0.0011	0.0066	0.0040	0.0028
I_2	0.0067	0.0170	0.0059	0.0095	0.0001
I_3	0.0207	0.0123	0.0360	0.0036	0.0000
I_4	0.0094	0.0149	0.0140	0.0114	0.0130
μ	0.0102	0.0121	0.0136	0.0106	0.0037
σ	0.0058	0.0057	0.0116	0.0075	0.0048
$\mu = 0.0100, \sigma = 0.0084$					

(b) Evaluation for $\sigma_c, \sigma_p = 0.05$ and $n_\theta = 20$.

Id	Hypothesis, $\Delta[\circledast_*]^{\ominus}$				
	H_0	H_1	H_2	H_3	H_4
I_0	0.0116	0.0299	0.0475	0.0507	0.0185
I_1	0.0413	0.0994	0.0287	0.0147	0.0291
I_2	0.0395	0.0053	0.0093	0.0127	0.0002
I_3	0.0537	0.0137	0.0537	0.2829	0.0000
I_4	0.0440	0.0902	0.0405	0.0491	0.0020
μ	0.0380	0.0477	0.0359	0.0820	0.0100
σ	0.0141	0.0394	0.0157	0.1017	0.0118
$\mu = 0.0427, \sigma = 0.0563$					

(c) Evaluation for $\sigma_c, \sigma_p = 0.10$ and $n_\theta = 10$.

Id	Hypothesis, $\Delta[\circledast_*]^{\ominus}$				
	H_0	H_1	H_2	H_3	H_4
I_0	0.0094	0.0219	0.2249	0.0390	0.0235
I_1	0.0391	0.0155	0.0311	0.0103	0.0252
I_2	0.0117	0.0059	0.0064	0.0321	0.0007
I_3	0.0650	0.0500	0.0240	0.3870	0.0000
I_4	0.0523	0.0998	0.0200	0.2524	0.0406
μ	0.0355	0.0386	0.0613	0.1442	0.0180
σ	0.0220	0.0339	0.0822	0.1498	0.0156
$\mu = 0.0595, \sigma = 0.0924$					

(d) Evaluation for $\sigma_c, \sigma_p = 0.10$ and $n_\theta = 20$.

Id	Hypothesis, $\Delta[\circledast_*]^{\ominus}$				
	H_0	H_1	H_2	H_3	H_4
I_0	0.2542	0.1102	0.0310	0.1427	0.1157
I_1	0.0226	0.2396	0.0731	0.0896	0.0003
I_2	0.0872	0.0806	0.0851	0.0487	0.0021
I_3	0.0842	0.0668	0.0688	0.1702	0.0767
I_4	0.1473	0.4717	0.1661	0.1764	0.1154
μ	0.1191	0.1938	0.0848	0.1255	0.0620
σ	0.0782	0.1518	0.0445	0.0491	0.0517
$\mu = 0.1171, \sigma = 0.0982$					

(e) Evaluation for $\sigma_c, \sigma_p = 0.25$ and $n_\theta = 10$.

Id	Hypothesis, $\Delta[\circledast_*]^{\ominus}$				
	H_0	H_1	H_2	H_3	H_4
I_0	0.1108	0.1154	0.0175	0.1561	0.1725
I_1	0.0301	0.1530	0.0761	0.0931	0.0085
I_2	0.0784	0.0146	0.0358	0.0469	0.0014
I_3	0.0098	0.0728	0.0742	0.1582	0.0026
I_4	0.1315	0.2415	0.0695	0.0942	0.0637
μ	0.0721	0.1195	0.0546	0.1097	0.0497
σ	0.0463	0.0764	0.0237	0.0423	0.0656
$\mu = 0.0811, \sigma = 0.0624$					

(f) Evaluation for $\sigma_c, \sigma_p = 0.25$ and $n_\theta = 20$.

Id	Hypothesis, $\Delta[\circledast_*]^{\ominus}$				
	H_0	H_1	H_2	H_3	H_4
I_0	0.1198	0.3590	0.5890	0.1357	0.3296
I_1	0.1111	0.2604	0.6792	0.0471	0.0072
I_2	0.1199	0.2686	0.7869	0.0417	0.0667
I_3	0.1852	0.1863	0.8134	0.0970	0.0044
I_4	0.2608	0.2785	0.4548	0.0491	0.2802
μ	0.1594	0.2706	0.6647	0.0741	0.1376
σ	0.0573	0.0549	0.1319	0.0367	0.1393
$\mu = 0.2613, \sigma = 0.2363$					

(g) Evaluation for $\sigma_c, \sigma_p = 0.50$ and $n_\theta = 10$.

Id	Hypothesis, $\Delta[\circledast_*]^{\ominus}$				
	H_0	H_1	H_2	H_3	H_4
I_0	0.0370	0.3463	0.7984	0.0917	0.3493
I_1	0.2121	0.1809	0.8057	0.0792	0.9170
I_2	0.1442	0.4411	0.6479	0.1276	0.0005
I_3	0.2743	0.2536	0.7995	0.0949	0.4984
I_4	0.0990	0.5428	0.5022	0.0866	0.1524
μ	0.1533	0.3529	0.7107	0.0960	0.3835
σ	0.0833	0.1291	0.1200	0.0167	0.3159
$\mu = 0.3393, \sigma = 0.2784$					

(h) Evaluation for $\sigma_c, \sigma_p = 0.50$ and $n_\theta = 20$.

Table 7.41: Results from Scenario VI, noisy optimization parameterized with SO(n) on hypothesis set $\mathrm{H}_3^{\ominus}\mathrm{SE}(2)$. Sensor and actuator noise ranging from $\sigma_c, \sigma_p = 0.05$ to $\sigma_c, \sigma_p = 0.50$ Commutativity delta $\Delta[\circledast_*]^{\ominus}$ per iteration I_0 to I_4. Mean and standard deviation of $\Delta[\circledast_*]^{\ominus}$ per hypothesis H_0 to H_4 and over all hypotheses.

the true one. Increasing the noise caused AgentLoop::Optimize to fail in doing so. Remarkably, the number of particles in the noisy scenarios did not affect the result too much.

7.5 Summary of Experiments

In this chapter, we showed that the sensorimotor map can be used as a tool for detecting the sensorimotor properties of the interaction between a mobile agent and its environment.

In Scenario I, we demonstrated that the two conflict functions developed in Chapter 6 can detect disorder in a given set of data points and that their return value is proportional to the degree of conflict introduced into these sets. Furthermore we identified a relation between the agent step size and the kernel widths of these functions, allowing us to directly link the conflict functions to the properties of the exploration patterns.

In Scenario II and Scenario III, we showed that the classification approach (see Algorithm 15) can be used to identify the true parameterization of the sensorimotor map from a set of hypotheses, based on a set of data points that were perceived through exploration of an unknown environment. This holds true for holonomic and nonholonomic systems in two- and in three-dimensional settings. Using the Nadaraya-Watson conflict function developed in Section 6.2.2.1, a high classification accuracy can be achieved in all experiments. The RBFN conflict function developed in Section 6.2.2.2, however, showed a considerably worse performance. Due to its inconsistent behavior we decided not to utilize it in further experiments on $SE(2)$. Considering the results using the Nadaraya-Watson conflict function, we identified a relation between the agent's step size, the exploration pattern, and the classification accuracy, which allowed us to select a suitable combination of these values that we used in the next experiments.

In Scenario V, we showed that the classification approach (see Algorithm 15) can achieve a high classification accuracy using both shorter exploration patterns and subsets of the available data points, showing potential for reducing computation time. We demonstrated that the classification accuracy of Algorithm 15 is robust to actuator and sensor noise. Compared to a classification baseline using a ground truth map of the environment created in Scenario IV, Algorithm 15 was shown to be more robust, where a comparable decline in classification accuracy started at a noise level two to four times as high. With respect to the data points we showed that occupancy data alone is sufficient for discovering the true hypothesis reasonably well. Disabling occupancy data allowed us to assess the impact of individual sensors on the performance of Algorithm 15, where the temperature sensor stood out as the sensor with the highest impact, followed by the light sensor and the range sensor. We identified the proximity sensor as a sensor with a negative impact on the classification accuracy, discouraging its usage.

In Scenario VI, we showed that the optimization approach (see Algorithm 16) can be used to identify a parameterization of the sensorimotor map that is comparable to the true hypothesis. We demonstrated that the performance of Algorithm 16 considerably depends on the selected search space. Utilizing a parameterization based on the special orthogonal group, Algorithm 16 was able to identify the true hypothesis. Utilizing a parameterization based on the general linear group, however, the performance of Algorithm 16 was shown to be considerably worse. In a final evaluation we demonstrated that Algorithm 16 can even be used in moderately noisy scenarios to identify a parameterization of the sensorimotor map that is comparable to the true

hypothesis. However, the noise level at which the performance of the optimization approach starts to decline was shown to be below the level the classification approach could work with, which was expected.

8 Conclusion

In this chapter we provide a brief summary of the work presented in this thesis. Afterwards, we will discuss further research directions.

8.1 Summary and Discussion

In this thesis, we developed a domain-independent, graph-based bootstrapping algorithm that allows a mobile agent to create a consistent geometric representation of its environment and its means of locomotion. The algorithm utilizes sequences of the agent's previous sensorimotor interaction with its environment while only requiring minimal a priori information about the agent's sensory and motor capabilities.

We established a mathematical foundation for representing sensorimotor systems, allowing us to formally describe the interactions of an agent with its environment. To that end we utilize manifolds, mathematical structures that are locally homeomorphic to the n-dimensional Euclidean space. More specifically, we use matrix Lie groups, smooth manifolds endowed with a group structure that model rigid transformations in Euclidean space. Based on this foundation, we developed the sensorimotor map, a domain-independent parameterizable graph-based structure, which requires minimal a priori knowledge of the agent's sensors and actuators to be constructed. This graph decomposes the agent's environment into a set of overlapping coordinate charts, where each of these charts corresponds to a local neighborhood on the manifold. As the coordinate charts are represented with respect to the agent's motor capabilities, transitions between local patches directly correspond to executable controls.

We related the sensorimotor map to the bootstrapping scenario. Given the history of past interactions between the agent and an environment with an unknown manifold structure, the aim is to determine its properties by finding a parameterization of the sensorimotor map that creates a consistent representation.

We motivated the notion of intrinsic conflict, a measurable property of disorder in the sensorimotor map, and developed two kernel based functions that allow us to estimate it. The first utilizes a modified version of the Nadaraya-Watson kernel estimator and the second utilizes a radial basis function network. Together with a suitable parameterization of the sensorimotor map itself, this allowed us to use the map in two bootstrapping algorithms: In the first one, we use the sensorimotor map as a classifier. Given a set of hypotheses that describe alternative configurations of the environment, the aim is to utilize the sensorimotor map to find the true hypothesis as the one with the least amount of intrinsic conflict. In the second one, we use the sensorimotor map as an objective function for a particle swarm optimization algorithm to determine the parameterization of the sensorimotor map with the minimal conflict.

To evaluate these approaches, we developed a simulation system that allows us to simulate the interaction of a virtual mobile agent with its environment. The agent is equipped with an actuator that can be freely parameterized to reflect different ways of how controls affect the agent's position and orientation. Feedback on the partial execution of controls allows the agent to assess whether portions of the environment are obstructed or free. In addition it can perceive properties of its environment using different types of sensors: a light sensor, a sensor that allows it to detect obstacles in close proximity, a range sensor that measures the distance to the next obstacle in the agent's line of sight, and a temperature sensor.

We showed that the bootstrapping classification approach is able to detect the true parameterization of the sensorimotor map with a high accuracy in various scenarios, where we considered both a holonomic and a nonholonomic case in two- and three-dimensional settings. In this scenarios, the sensorimotor map was created from the sensorimotor chain, a subjective history of the agent's sensorimotor interaction with its environment. It consists of an alternating sequence of sensory inputs and motor controls, recorded while the agent explored its environment. It was shown that the performance of the bootstrapping algorithm was mainly dependent on two parameters that describe the agent's exploration strategy: the agent's step size and the type of exploration pattern.

The step size determines the magnitude of each of the agent's controls and thus has a direct impact on the size of the explored area in one step. It was shown that a larger step size considerably improves the classification accuracy. This is due to the way parameterizations of the sensorimotor map relate the agent's controls to the Lie group representation of its configuration space. Each parameterization associates a specific set of vector fields to the agent's controls. Different parameterizations thus differ with respect to the amount their corresponding vector fields commute, a relation expressed by the Lie bracket. For smaller step sizes the effect of non-commuting vector fields is small, causing the sensorimotor projection sets of the associated sensorimotor maps to look similar. Thus, for smaller step sizes, sensorimotor maps created from different parameterizations look too similar for the conflict function to detect a substantial difference.

Exploration patterns consist of a sequence of controls and allow the agent to pursue two substantially different strategies: One is to regularly return to the agent's starting location, exploring the same local patch of the configuration space multiple times, i.e., creating fewer sensorimotor neighborhoods with more data points. The other is to perform a random walk, exploring a larger area of the configuration space, i.e., creating more sensorimotor neighborhoods but with fewer

data points each. It was shown that the second type of pattern performs considerably better in almost all scenarios. This was an unexpected result, as with respect to the conflict calculation, we assumed that having more data points available in each sensorimotor neighborhood would increase the potential for intrinsic conflict and, thus, would improve the performance of the bootstrapping algorithm. As the simulation environment was surrounded by boundaries, the agent naturally was constricted to this local region. Thus, even when we utilize an exploration pattern that does not force the agent to return to its starting location, the environment itself imposed an 'extrinsic' returning behavior, inevitably causing the agent to measure the same local patch of the environment multiple times.

We showed that the bootstrapping classification algorithm, utilizing the sensorimotor map, is more robust to actuator and sensor noise than a comparable approach using a ground truth map of the environment. This stems from differences in the way that actuator noise affects these two representations: The ground truth map represents the agents's position with respect to a single global reference frame with its origin at the agent's initial position. Successive execution of noisy controls inevitably causes an odometry error to accumulate, continuously decreasing the usefulness of the map as a foundation for calculating the intrinsic conflict, due to the fact that the true and the assumed position diverge more and more. The sensorimotor map, however, represents the environment as a set of local neighborhoods, where each neighborhood establishes a local reference frame that represents a local, bounded patch of the environment. This spatial subdivision mitigates the odometry error, as in each neighborhood only a local, spatially related subset of the data points from the sensorimotor chain is represented.

We showed that the performance of the bootstrapping classification algorithm is considerably affected by the type of sensor measurements it can utilize and dependent on whether or not information on empty and occupied configurations is available. In this regard, we showed that the bootstrapping classification approach is able to detect the true parameterization of the sensorimotor map with a high accuracy, even when utilizing occupancy data alone. This is due to the fact that the occupancy function divides the entire environment into occupied or free regions that are connected. Introducing even a small amount of disorder by slightly altering the parameterization of the sensorimotor map inevitably causes these blocks to 'unravel' and 'fray': Single empty configurations are projected into occupied regions and single occupied configurations are projected into empty regions. This rapidly increases the intrinsic conflict in each neighborhood.

Excluding the utilization of occupancy data allowed us to evaluate and assess the impact of individual sensors on the performance of the bootstrapping algorithm. We showed that the temperature sensor had the highest impact on the classification accuracy, followed by the light sensor and the range sensor. The proximity sensor, however, was shown to have a negative impact on the classification accuracy. Temperature sensor measurements are generated by applying a Perlin noise function to the coordinates of the environment, assigning a nonlinear, smoothly varying function to the configuration space. Even small changes in the parameterization of the sensorimotor map introduce disorder in this smooth structure, consequently increasing the detectable intrinsic conflict. Light and range sensor measurements behave differently, as they vary more gradually. Thus, the impact of different parameterizations of the sensorimotor map is less distinctive, making the induced disorder less easy to detect. Proximity sensor measurements are almost zero for all configurations, with the exception being configurations where the agent is

in close proximity to a boundary. This corresponds to a thin non-zero outline around obstacles which, as it was shown, does not provide enough information to perform a reliable classification.

We showed that the bootstrapping optimization approach is able to find a parameterization of the sensorimotor map similar to the true parameterization, utilizing the sensorimotor map as an objective function in conjunction with a particle swarm optimization algorithm as an instance of a nonparametric optimization technique. The results were dependent on the search space. The bootstrapping optimization algorithm was able to identify suitable parameterizations of the sensorimotor map in scenarios where the search space was parameterized by the special orthogonal group. In scenarios where the search space was parameterized by the general linear group, the performance was considerably worse. The main difference between these two search space parameterizations is, next to the differences in their dimensionality, their ability to express certain types of projection matrices. The special orthogonal group only allows to encode projection matrices that correspond to rotations. These parameterizations rotate the coordinate system the agent's controls are expressed in and map it to the tangent space of the hypothesis Lie group. The orthogonal group allows us to encode projection matrices that correspond to arbitrary linear transformations, e.g. scaling and skewing. This descriptive freedom, however, is counterbalanced by the size of the search space, which, in turn, makes the optimization problem considerable harder.

We showed that using the search space given by the special orthogonal group, the bootstrapping optimization algorithm was able to identify suitable parameterizations of the sensorimotor map, even in moderately noisy scenarios. As expected, the noise level at which its performance declined was shown to be below the level of the bootstrapping classification algorithm.

8.2 Outlook

Considering future research directions, we identify several interesting topics related to the sensorimotor map directly or to its utilization in the bootstrapping context.

Currently, the relation between the agent's controls and the tangent space of the Lie group, which is used to represent the configuration space, is modeled as a linear transformation, encoded in the projection matrix. As there exists a direct mapping between controls and group elements, each control directly corresponds to a distinct vector field on the configuration space Lie group. While this representation allows us to model a variety of mobile systems, certain control types can not be represented in this way. A prime example would be the Ackerman steering configuration used in cars [Siegwart and Nourbakhsh, 2004, pp. 37ff.]. Here, turning the steering wheel changes the position of the front wheels which then changes the way the car behaves when driving forward or backward. Rather than directly changing the agent's pose, one element of the control vector changes the transformation the other one will induce. With respect to the way we model the sensorimotor map, this would correspond to having different types of controls: direct controls that are mapped to the Lie group as currently implemented, and indirect controls that allow the agent to manipulate this mapping. Integrating the layered sensorimotor map into the bootstrapping process would be a challenging task, as it would add an additional level of complexity by introducing an interdependent set of parameters to be detected.

Another possible direction would be to utilize the sensorimotor map for path planning. This would require the agent to evaluate the sensorimotor map with respect to certain policies, allowing the agent to assign risk and reward values to distinct configurations, based on their associated samples and obstructions. As each sensorimotor neighborhood is represented in the agent's control reference frame, identifying a beneficial configuration automatically provides the agent with the control required to reach it. While selecting the next immediate control is therefore a straightforward task, calculating a sequence of control, i.e., a path through the configuration space is more challenging: For every executed control the agent may changes its local reference frame, potentially transitions from one neighborhood to another, where for each transition the path eventually would have to be updated according to the new information.

A related topic is the explicit modeling of noise and uncertainty. Though the algorithm was shown to be inherently robust to actuator and sensor noise, performing bootstrapping reliably under these conditions would require methods to explicitly model the effect of noisy actuator controls and sensor measurements. A computationally expensive ad-hoc solution would be to combine the concept of a particle filter (PF) [Thrun et al., 2005, Chapter 4.3] with the sensorimotor map. Each particle would contain a separate instance of the sensorimotor map, where the weight of the particles would have to be based on the likelihood of the latest sample and obstruction data, given the current state of the map. Calculating this likelihood would require corresponding modifications to the sensorimotor neighborhoods and the regression algorithms that are currently used to estimate the sample and occupancy functions. Each hypothesis in the bootstrapping classification scenario and likewise each PSO particle in the bootstrapping optimization scenario would contain a separate PF instance. Especially in the optimization scenario, rebuilding the sensorimotor map for each PSO and thus each PF particle would be expensive and likely impractical. A potential alternative to reduce the computational cost would be to utilize concepts like graph-based SLAM [Grisetti et al., 2010]. Investigating the implications this would have on the structure of the sensorimotor map and the underlying graph-structure, would be a challenging task as well.

A question related to the bootstrapping scenario is, whether we can increase the performance of the system by modifying the way that the agent explores its environment. The exploration strategy currently used is inspired by the concept of motor babbling (see Section 2.2.3). Thus the exploration patterns are predefined, randomly generated sequences of controls. In order to adapt to deviations, introduced by colliding with obstacles, the agent is able to perform minor adjustments to single controls. However, the overall sequence is not modified in any way. Consequently, information on the current state of the bootstrapping algorithm is not incorporated into the agent's behavior. An extension would be to establish a relation between the selection of motor controls and the current assumption the agent has with respect to the structure of its environment. Thus, instead of choosing motor controls randomly, the agent would select controls in such a way as to actively verify or contradict certain hypotheses. A requirement for such an active exploration behavior (see Section 2.2.4), however, is to specify the 'potential intrinsic conflict' of a given motor control, i.e., the potential disorder a certain sample or obstruction introduces into the sensorimotor map. Utilizing this function, the agent could calculate the information gain [Schill, 1997], or more specifically, the 'conflict gain' as the expected increase of classification-relevant information contained in the sensorimotor map. Approaches that demon-

strate the use of information gain for active exploration in three-dimensional Euclidean space have, for example, been developed in [Nakath et al., 2016, Nakath et al., 2019], where this concept is applied to the task of long-term planning of a spacecraft's orientation during interplanetary flight, i.e., a hybrid localization, and a mapping scenario of a spacecraft orbiting an asteroid, respectively. Adapting the latter concept to the bootstrapping scenario would correspond to a mechanism that allows the agent to deliberatively choose either of two types of controls: non-commutative, i.e., hypothesis invariant, actions to bolster its localization hypothesis or commutative and hypothesis-specific actions, that increase the potential intrinsic conflict in the sensorimotor map.

A final question considers the bootstrapping scenario as a whole: From a mathematical point of view, the bootstrapping approach is closely related to the concept of manifold learning: We constructed the Lie group that defines the agent's configuration space as an atlas of overlapping neighborhoods, based on executed trajectories that are composed of piecewise geodesics. In this regard, the sensorimotor map provides a graph-based structure that defines a parameterizable homeomorphism, i.e., we established a formalism that allows us to modify the bidirectional mapping from neighborhoods on the Lie group, to local patches in the associated Euclidean space. The question is to which extent information on Lie groups, the calculation of the group operation and the calculation of commutativity can be assumed to be a priori knowledge? Though we did not provide any information on sensors and actuators to the system, we, however, endowed the system with certain mathematical structures and the ability to perform certain mathematical operations on them. The system utilizes Lie groups as a foundation for representing the configuration space and consequently the related mathematical operations, i.e., the group operation, given as the matrix multiplication, and the mapping between the group and its tangent space, given by the matrix exponential and the matrix logarithm. Thus, the questions are, to which extent this reduction of a priori knowledge can be considered 'minimal' and to which extent it can be considered 'sufficient'. To answer the first question, we have to investigate, whether certain aspects of Lie groups and the related could be even more generalized, allowing us to further minimize the information provided to the agent. To answer the second question, the next step would be to apply the bootstrapping algorithm to a real-world robotic system and evaluate its performance. Ultimately the goal would be to be able to utilize the algorithm developed in this work to endow arbitrary robotic systems with the ability to develop their own intrinsic representation of their environment, making them independent of the requirement for humans to be involved in the bootstrapping process. As stated in [Censi and Murray, 2015] it *"would be extremely convenient if we could just attach any sensor to a robot, and the robot could learn how to use it, without tedious programming"*. The approach presented in this work, may provide a small contribution to this goal.

A Prominent Lie Groups

This appendix provides a brief overview of prominent matrix Lie groups and their corresponding Lie algebras. The matrix Lie groups presented in the following sections are special as they represent positions and orientations in Euclidean space.
The content of this appendix is based on [Choset et al., 2005, Chapter 3.5.1] and [Bloch et al., 2007, Chapters 1.4 and 2.8].

A.1 Special Orthogonal Group SO(n)

The special orthogonal group of dimension n describes the rotation of objects in n-dimensional Euclidean space [Bloch et al., 2007, p. 100]. Formally it consists of all $n \times n$ matrices for which holds that

$$SO(n) = \{ A \in \mathbb{R}^{n \times n} \mid A^T A = A A^T = I_{n \times n}, \det(A) = 1 \}. \tag{A.1}$$

The Lie algebra of $SO(n)$ consists of all skew-symmetric matrices, thus

$$\mathfrak{so}(n) = \{ A \in \mathbb{R}^{n \times n} \mid A^T = -A \}. \tag{A.2}$$

Each element $\mathfrak{g} \in \mathfrak{so}(n)$ can be represented as a linear combination of its generators. The number of generators, corresponding to the degree of freedom, is given as the number of unique pairs of dimensions, calculated by the binomial coefficient as

$$\mathrm{DoF}_{SO(n)} = \binom{n}{2} = \frac{n!}{k!(n-k)!} \quad \text{given} \quad n > 1. \tag{A.3}$$

A.1.1 Two-Dimensional Rotation Group SO(2)

The rotation group $SO(2)$ is a 1-dimensional Lie group and represents rotations in $\mathbb{R}^2$ [Choset et al., 2005, pp. 61–62]. Its Lie algebra $\mathfrak{so}(2)$ has a single generator given as

$$\hat{\mathfrak{g}} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}, \tag{A.4}$$

which corresponds to rotations around the origin in the x/y-plane. Elements $\mathfrak{g} \in \mathfrak{so}(2)$ are created by multiplying the generator $\hat{\mathfrak{g}}$ with a scalar $r \in \mathbb{R}$, thus

$$\mathfrak{so}(2) = \{\, r\,\hat{\mathfrak{g}} \,|\, r \in \mathbb{R} \,\}. \tag{A.5}$$

Elements $g \in SO(2)$ are created by applying the matrix exponential to an element $\mathfrak{g} \in \mathfrak{so}(2)$, with

$$g = \exp(\, r\hat{\mathfrak{g}}\,) = \exp \begin{bmatrix} 0 & -r \\ r & 0 \end{bmatrix} = \begin{bmatrix} \cos r & -\sin r \\ \sin r & \cos r \end{bmatrix}, \tag{A.6}$$

where r corresponds to the rotation angle.

A.1.2 Three-Dimensional Rotation Group SO(3)

The rotation group $SO(3)$ is a 3-dimensional Lie group and represents rotations in $\mathbb{R}^3$ [Choset et al., 2005, pp. 61–62]. Its Lie algebra $\mathfrak{so}(3)$ has three generators given as

$$\hat{\mathfrak{g}}_1 = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \hat{\mathfrak{g}}_2 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix}, \hat{\mathfrak{g}}_3 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}, \tag{A.7}$$

which correspond to rotations around the origin in the x/y, x/z, and y/z plane, respectively. Elements $\mathfrak{g} \in \mathfrak{so}(3)$ are created as linear combinations of the generators $\hat{\mathfrak{g}}_i$ with scalars $r_i \in \mathbb{R}$, thus

$$\mathfrak{so}(3) = \{\, r_1\hat{\mathfrak{g}}_1 + r_2\hat{\mathfrak{g}}_2 + r_3\hat{\mathfrak{g}}_3 \,|\, r_1, r_2, r_3 \in \mathbb{R} \,\}. \tag{A.8}$$

Elements $g \in SO(3)$ are created by applying the matrix exponential to an element $\mathfrak{g} \in \mathfrak{so}(3)$, with

$$g = \exp(\, r_1\hat{\mathfrak{g}}_1 + r_2\hat{\mathfrak{g}}_2 + r_3\hat{\mathfrak{g}}_3\,) = \exp \begin{bmatrix} 0 & -r_1 & r_2 \\ r_1 & 0 & -r_3 \\ -r_2 & r_3 & 0 \end{bmatrix}. \tag{A.9}$$

A.2 Special Euclidean Group SE(n)

The special Euclidean group describes rigid transformations in n-dimensional Euclidean space [Choset et al., 2005, p. 62]. Elements from $SE(n)$ can be represented as $(n+1)\times(n+1)$ matrices composed from an element from $SO(n)$, which represents the rotation component and an element from $\mathbb{R}^n$, which represents the translation component, thus

$$SE(n) = \left\{ \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \middle| R \in SO(n),\, T \in \mathbb{R}^n \right\} = SO(n) \times \mathbb{R}^n. \tag{A.10}$$

Each element $g \in \mathfrak{se}(n)$ can be represented as a linear combination of its generators, where the number of generators, and thus the degree of freedom, is equal to $\frac{1}{2}(n^2 - n) + n$, given $n > 1$.

A.2.1 Three-Dimensional Special Euclidean Group SE(2)

The special Euclidean group $SE(2)$ is a 3-dimensional Lie group and represents rigid transformations in $\mathbb{R}^2$ [Choset et al., 2005, p. 62],[Bloch et al., 2007, p. 16]. Its Lie algebra $\mathfrak{se}(2)$ has three generators given as

$$\hat{\mathfrak{g}}_1 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \hat{\mathfrak{g}}_2 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}, \hat{\mathfrak{g}}_3 = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \tag{A.11}$$

where $\hat{\mathfrak{g}}_1$ and $\hat{\mathfrak{g}}_2$ correspond to translations along the x and y axis, respectively, and $\hat{\mathfrak{g}}_3$ corresponds to a rotations in the x/y plane. Elements $\mathfrak{g} \in \mathfrak{se}(2)$ are created as linear combinations of the generators $\hat{\mathfrak{g}}_i$ with scalars $r_i \in \mathbb{R}$, thus

$$\mathfrak{se}(2) = \{\, r_1\hat{\mathfrak{g}}_1 + r_2\hat{\mathfrak{g}}_2 + r_3\hat{\mathfrak{g}}_2 \,|\, r_1, r_2, r_3 \in \mathbb{R} \,\}. \tag{A.12}$$

Elements $g \in SE(2)$ are created by applying the matrix exponential to an element $\mathfrak{g} \in \mathfrak{se}(2)$, with

$$\exp(\, r_1\hat{\mathfrak{g}}_1 + r_2\hat{\mathfrak{g}}_2 + r_3\hat{\mathfrak{g}}_3 \,) = \exp \begin{bmatrix} 0 & -r_3 & r_1 \\ r_3 & 0 & r_2 \\ 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} \cos r_3 & -\sin r_3 & r_1 \\ \sin r_3 & \cos r_3 & r_2 \\ 0 & 0 & 1 \end{bmatrix}. \tag{A.13}$$

A.2.2 Six-Dimensional Special Euclidean Group SE(3)

The special Euclidean group $SE(3)$ is a 6-dimensional Lie group and represents rigid transformations in $\mathbb{R}^3$ [Choset et al., 2005, p. 62], [Bloch et al., 2007, p. 101]. Its Lie algebra $\mathfrak{se}(3)$ has three generators given as

$$\hat{\mathfrak{g}}_1 = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad \hat{\mathfrak{g}}_2 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad \hat{\mathfrak{g}}_3 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix},$$

$$\hat{\mathfrak{g}}_4 = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \hat{\mathfrak{g}}_5 = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \hat{\mathfrak{g}}_6 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix},$$

where $\hat{\mathfrak{g}}_1, \hat{\mathfrak{g}}_2$ and $\hat{\mathfrak{g}}_3$ correspond to translations along the x, y and z axis, while $\hat{\mathfrak{g}}_4, \hat{\mathfrak{g}}_5$ and $\hat{\mathfrak{g}}_6$ correspond to rotations in the x/y, x/z, and y/z plane, respectively. Elements $\mathfrak{g} \in \mathfrak{se}(3)$ are created as linear combinations of the generators $\hat{\mathfrak{g}}_i$ with scalars $r_i \in \mathbb{R}$, thus

$$\mathfrak{se}(3) = \{ r_1\hat{\mathfrak{g}}_1 + r_2\hat{\mathfrak{g}}_2 + r_3\hat{\mathfrak{g}}_3 \ + r_4\hat{\mathfrak{g}}_4 + r_5\hat{\mathfrak{g}}_5 + r_5\hat{\mathfrak{g}}_6 \,|\, r_1, \cdots, r_6 \in \mathbb{R} \}. \tag{A.14}$$

Elements $g \in SE(3)$ are created by applying the matrix exponential to an element $\mathfrak{g} \in \mathfrak{se}(3)$, with

$$\begin{aligned} g &= \exp(r_1\hat{\mathfrak{g}}_1 + r_2\hat{\mathfrak{g}}_2 + r_3\hat{\mathfrak{g}}_3 + r_4\hat{\mathfrak{g}}_4 + r_5\hat{\mathfrak{g}}_5 + r_5\hat{\mathfrak{g}}_6) \\ &= \exp \begin{bmatrix} 0 & -r_4 & r_5 & r_1 \\ r_4 & 0 & -r_6 & r_2 \\ -r_5 & r_6 & 0 & r_3 \\ 0 & 0 & 0 & 0 \end{bmatrix}. \end{aligned} \tag{A.15}$$

B

Mobile Robot Example

As an example for the effect of different parameterizations of the transition function α (see Eq. (4.11)), we revisit the wheeled robot with two degrees of freedom moving in the plane discussed in Section 3.7, depicted in Fig. 3.10b. This robot is a nonholonomic system as its control space C resembles $\mathbb{R}^2$ where the configuration space is the three-dimensional matrix Lie group $SE(2)$, thus the extended control space C_* corresponds to $\mathbb{R}^3$. We assume that the robot is equipped with a slip/skid steering configuration [Siegwart and Nourbakhsh, 2004, Chapter 2.3.2.3].

B.1 Example I

Consider the mapping matrix T_* and the projection matrix $T_\cup$

$$T_* = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \quad \text{and} \quad T_\cup = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}. \tag{B.1}$$

Given the control vector $c_\odot = [\, c_1,\, c_2 \,]^T$ with $c_\odot \in C_\odot \subseteq C$, applying $(\tau_\cup \circ \tau_*)(c_\odot)$ yields

$$T_\cup T_* c = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ 0 \end{bmatrix} = \begin{bmatrix} c_1 \\ 0 \\ c_2 \end{bmatrix} \tag{B.2}$$

which corresponds to the Lie algebra element

$$\mathfrak{g}_{c_\odot} = c_1 \hat{\mathfrak{g}}_1 + c_2 \hat{\mathfrak{g}}_3. \tag{B.3}$$

The first element c_1 of a control vector $[c_1, c_2]^T \in C$ is mapped to the generator $\hat{\mathfrak{g}}_1$ which corresponds to translation in x-direction, the second element c_2 is mapped to the generator $\hat{\mathfrak{g}}_3$

which corresponds to rotation in the x/y-plane. Therefore, the matrix $T_{\circlearrowleft}$ maps controls from $C_{\odot}$ in such a way that scalar multiples of basis vectors in $C_{\odot}$ correspond to individually translating forward and backward as well as individually rotating on the spot. This corresponds to a mobile entity with some kind of abstraction layer that accounts for individually rotating its left and right wheels based on the controls provided.

B.2 Example II

In a second example we set the projection matrix $T_{\circlearrowleft}$ to be

$$T_{\circlearrowleft} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & -1 \\ 1 & -1 & 0 \end{bmatrix}. \tag{B.4}$$

Given the control vector $c_{\odot} = [\, c_1,\, c_2 \,]^T$ with $c_{\odot} \in C_{\odot} \subseteq C$, applying $(\tau_{\circlearrowleft} \circ \tau_*)(c_{\odot})$ yields

$$T_{\circlearrowleft} T_* c = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & -1 \\ 1 & -1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & -1 \\ 1 & -1 & 0 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ 0 \end{bmatrix} = \begin{bmatrix} c_1 + c_2 \\ 0 \\ c_1 - c_2 \end{bmatrix}, \tag{B.5}$$

which corresponds to the Lie algebra element

$$\mathfrak{g}_{c_{\odot}} = (c_1 + c_2)\, \hat{\mathfrak{g}}_1 + (c_1 - c_2)\, \hat{\mathfrak{g}}_3. \tag{B.6}$$

In this case, the first element c_1 of a control vector $[c_1, c_2]^T \in C$ is mapped to both generators $\hat{\mathfrak{g}}_1$ and $\hat{\mathfrak{g}}_3$ which corresponds to a combined transform consisting of a translation forward and a counter-clockwise rotation in the x/y-plane. Correspondingly the second element c_2 is mapped to both generators $\hat{\mathfrak{g}}_1$ and $\hat{\mathfrak{g}}_3$ as well. However, the sign change indicates that it corresponds to a combined action consisting of a forward translation and a clockwise rotation. Therefore, this matrix $T_{\circlearrowleft}$ maps controls from $C_{\odot}$ in such a way that scalar multiples of basis vectors in $C_{\odot}$ correspond to combined translations and rotations, i.e. individual controls correspond to curved paths in Q^n. However, straight transformation along the x-axis is possible by selecting a control $c'_{\odot} = [\, r,\, r \,]^T$ with $r \in \mathbb{R}$ which according to Eq. (B.6) yields

$$\mathfrak{g}_{c'_{\odot}} = (r + r)\, \hat{\mathfrak{g}}_1 + (r - r)\, \hat{\mathfrak{g}}_3 = 2r\hat{\mathfrak{g}}_1, \tag{B.7}$$

where the correspoding group element, given by the exponential map, is

$$\exp(\mathfrak{g}_{c'_{\odot}}) = \exp(2r\hat{\mathfrak{g}}_1) = \begin{bmatrix} 1 & 0 & 2r \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{B.8}$$

which is a matrix representing a translation of $2r$ along the x-axis. In the same fashion rotation on the spot is possible by selecting a control $c''_{\odot} = [\, r,\, -r \,]^T$ with $r \in \mathbb{R}$ which according to Eq. (B.6) yields

$$\mathfrak{g}_{c''_{\odot}} = (r - r)\, \hat{\mathfrak{g}}_1 + (r + r)\, \hat{\mathfrak{g}}_3 = 2r\hat{\mathfrak{g}}_3 \tag{B.9}$$

where the corresponding group element, given by the exponential map, is

$$\exp(\mathfrak{g}_{c''_{\odot}}) = \exp(2r\hat{\mathfrak{g}}_3) = \begin{bmatrix} cos(2r) & -sin(2r) & 0 \\ sin(2r) & cos(2r) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{B.10}$$

which is a matrix representing a $2r$ radians rotation in the x/y-plane accordingly. In this second example, the individual components of the control vector correspond to separately controlling the right (c_1-component) and the left (c_2-component) wheel.

C
List of Hypotheses

This appendix contains all hypothesis sets that have been utilized in the experiments conducted in Chapter 7. Each set contains a number of hypotheses, where for each hypothesis the weights vector and the projection matrix are given. In Appendices C.1 to C.3 the holonomic hypotheses for the matrix Lie group $SE(2)$ are given, while Appendices C.4 to C.6 depict the nonholonomic hypotheses for this Lie group. In Appendix C.7 the holonomic hypotheses for the matrix Lie group $SE(3)$ are given, where correspondingly Appendix C.8 contains the nonholonomic hypotheses for $SE(3)$.

C.1 Holonomic Hypotheses $H_1^{\oplus}SE(2)$ for $SE(2)$

Id	Weights	Projection Matrix $T_{\circlearrowleft}$
H_0	[0.5, 1.0, 1.5]	$\begin{bmatrix} 0.500 & 0.000 & 0.000 \\ 0.000 & 0.707 & -1.061 \\ 0.000 & 0.707 & 1.061 \end{bmatrix}$
H_1	[0.5, 1.0, 1.5]	$\begin{bmatrix} 0.500 & 0.000 & 0.000 \\ 0.000 & 1.000 & 0.000 \\ 0.000 & 0.000 & 1.500 \end{bmatrix}$
H_2	[0.5, 1.5, 1.0]	$\begin{bmatrix} 0.500 & 0.000 & 0.000 \\ 0.000 & 1.061 & -0.707 \\ 0.000 & 1.061 & 0.707 \end{bmatrix}$
H_3	[0.5, 1.5, 1.0]	$\begin{bmatrix} 0.500 & 0.000 & 0.000 \\ 0.000 & 1.500 & 0.000 \\ 0.000 & 0.000 & 1.000 \end{bmatrix}$
H_4	[1.0, 0.5, 1.5]	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 0.354 & -1.061 \\ 0.000 & 0.354 & 1.061 \end{bmatrix}$
H_5	[1.0, 0.5, 1.5]	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 0.500 & 0.000 \\ 0.000 & 0.000 & 1.500 \end{bmatrix}$
H_6	[1.0, 1.0, 1.0]	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 0.707 & -0.707 \\ 0.000 & 0.707 & 0.707 \end{bmatrix}$

Id	Weights	Projection Matrix $T_{\circlearrowleft}$
H_7	[1.0, 1.0, 1.0]	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 1.000 & 0.000 \\ 0.000 & 0.000 & 1.000 \end{bmatrix}$
H_8	[1.0, 1.5, 0.5]	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 1.061 & -0.354 \\ 0.000 & 1.061 & 0.354 \end{bmatrix}$
H_9	[1.0, 1.5, 0.5]	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 1.500 & 0.000 \\ 0.000 & 0.000 & 0.500 \end{bmatrix}$
H_{10}	[1.5, 0.5, 1.0]	$\begin{bmatrix} 1.500 & 0.000 & 0.000 \\ 0.000 & 0.354 & -0.707 \\ 0.000 & 0.354 & 0.707 \end{bmatrix}$
H_{11}	[1.5, 0.5, 1.0]	$\begin{bmatrix} 1.500 & 0.000 & 0.000 \\ 0.000 & 0.500 & 0.000 \\ 0.000 & 0.000 & 1.000 \end{bmatrix}$
H_{12}	[1.5, 1.0, 0.5]	$\begin{bmatrix} 1.500 & 0.000 & 0.000 \\ 0.000 & 0.707 & -0.354 \\ 0.000 & 0.707 & 0.354 \end{bmatrix}$
H_{13}	[1.5, 1.0, 0.5]	$\begin{bmatrix} 1.500 & 0.000 & 0.000 \\ 0.000 & 1.000 & 0.000 \\ 0.000 & 0.000 & 0.500 \end{bmatrix}$

Table C.1: Weights and projection matrices for $H_1^{\oplus}SE(2)$ indexed by hypothesis id. Hypothesis H_i created as tuple of SE(2), the projection matrix $T_{\circlearrowleft}$ and its inverse $T_{\circlearrowleft}^{-1} = T_{\circlearrowleft}$.

C.2 Holonomic Hypotheses Set $H_2^{\oplus}SE(2)$ on $SE(2)$

Id	Weights	Projection Matrix $T_{\circlearrowleft}$
H_0	[1.0, 1.0, 1.0]	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 0.981 & -0.195 \\ 0.000 & 0.195 & 0.981 \end{bmatrix}$
H_1	[1.0, 1.0, 1.0]	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 0.924 & -0.383 \\ 0.000 & 0.383 & 0.924 \end{bmatrix}$
H_2	[1.0, 1.0, 1.0]	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 0.831 & -0.556 \\ 0.000 & 0.556 & 0.831 \end{bmatrix}$

Id	Weights	Projection Matrix $T_{\circlearrowleft}$
H_3	[1.0, 1.0, 1.0]	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 0.707 & -0.707 \\ 0.000 & 0.707 & 0.707 \end{bmatrix}$
H_4	[1.0, 1.0, 1.0]	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 1.000 & 0.000 \\ 0.000 & 0.000 & 1.000 \end{bmatrix}$

Table C.2: Weights and projection matrices for $H_2^{\oplus}SE(2)$ indexed by hypothesis id. Hypothesis H_i created as tuple of SE(2), the projection matrix $T_{\circlearrowleft}$ and its inverse $T_{\circlearrowleft}^{-1} = T_{\circlearrowleft}$.

C.3 Holonomic Hypotheses Set $H_3^{\oplus}SE(2)$ on $SE(2)$

Id	Weights	Projection Matrix $T_{\circlearrowleft}$
H_0	[1.0, 1.0, 1.0]	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 0.924 & -0.383 \\ 0.000 & 0.383 & 0.924 \end{bmatrix}$
H_1	[1.0, 1.0, 1.0]	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 0.707 & -0.707 \\ 0.000 & 0.707 & 0.707 \end{bmatrix}$

Id	Weights	Projection Matrix $T_{\circlearrowleft}$
H_2	[1.0, 1.0, 1.0]	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 1.000 & 0.000 \\ 0.000 & 0.000 & 1.000 \end{bmatrix}$

Table C.3: Weights and projection matrices for $H_3^{\oplus}SE(2)$ indexed by hypothesis id. Hypothesis H_i created as tuple of SE(2), the projection matrix $T_{\circlearrowleft}$ and its inverse $T_{\circlearrowleft}^{-1} = T_{\circlearrowleft}$.

C.4 Nonholonomic Hypotheses Set $H_1^{\ominus}$SE(2) on SE(2)

Id	Weights	Projection Matrix $T_{\circlearrowleft}$
H_0	[0.5, 1.5]	$\begin{bmatrix} 0.500 & 0.000 & 0.000 \\ 0.000 & 1.061 & -0.707 \\ 0.000 & 1.061 & 0.707 \end{bmatrix}$
H_1	[0.5, 1.5]	$\begin{bmatrix} 0.500 & 0.000 & 0.000 \\ 0.000 & 1.500 & 0.000 \\ 0.000 & 0.000 & 1.000 \end{bmatrix}$
H_2	[0.5, 1.5]	$\begin{bmatrix} 0.000 & 0.000 & 1.000 \\ 0.500 & 0.000 & 0.000 \\ 0.000 & 1.500 & 0.000 \end{bmatrix}$
H_3	[1.0, 1.0]	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 0.707 & -0.707 \\ 0.000 & 0.707 & 0.707 \end{bmatrix}$
H_4	[1.0, 1.0]	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 1.000 & 0.000 \\ 0.000 & 0.000 & 1.000 \end{bmatrix}$
H_5	[1.0, 1.0]	$\begin{bmatrix} 0.000 & 0.000 & 1.000 \\ 1.000 & 0.000 & 0.000 \\ 0.000 & 1.000 & 0.000 \end{bmatrix}$
H_6	[1.5, 0.5]	$\begin{bmatrix} 1.500 & 0.000 & 0.000 \\ 0.000 & 0.354 & -0.707 \\ 0.000 & 0.354 & 0.707 \end{bmatrix}$
H_7	[1.5, 0.5]	$\begin{bmatrix} 1.500 & 0.000 & 0.000 \\ 0.000 & 0.500 & 0.000 \\ 0.000 & 0.000 & 1.000 \end{bmatrix}$
H_8	[1.5, 0.5]	$\begin{bmatrix} 0.000 & 0.000 & 1.000 \\ 1.500 & 0.000 & 0.000 \\ 0.000 & 0.500 & 0.000 \end{bmatrix}$

Table C.4: Weights and projection matrices for $H_1^{\ominus}$SE(2) indexed by hypothesis id. Hypothesis H_i created as tuple of SE(2), the projection matrix $T_{\circlearrowleft}$ and its inverse $T_{\circlearrowleft}^{-1} = T_{\circlearrowleft}$.

C.5 Nonholonomic Hypotheses Set $H_2^{\ominus}$SE(2) on SE(2)

Id	Weights	Projection Matrix $T_{\circlearrowleft}$
H_0	[1.0, 1.0]	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 0.981 & -0.195 \\ 0.000 & 0.195 & 0.981 \end{bmatrix}$
H_1	[1.0, 1.0]	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 0.924 & -0.383 \\ 0.000 & 0.383 & 0.924 \end{bmatrix}$
H_2	[1.0, 1.0]	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 0.831 & -0.556 \\ 0.000 & 0.556 & 0.831 \end{bmatrix}$
H_3	[1.0, 1.0]	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 0.707 & -0.707 \\ 0.000 & 0.707 & 0.707 \end{bmatrix}$
H_4	[1.0, 1.0]	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 0.556 & -0.831 \\ 0.000 & 0.831 & 0.556 \end{bmatrix}$
H_5	[1.0, 1.0]	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 0.383 & -0.924 \\ 0.000 & 0.924 & 0.383 \end{bmatrix}$
H_6	[1.0, 1.0]	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 0.195 & -0.981 \\ 0.000 & 0.981 & 0.195 \end{bmatrix}$
H_7	[1.0, 1.0]	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 1.000 & 0.000 \\ 0.000 & 0.000 & 1.000 \end{bmatrix}$
H_8	[1.0, 1.0]	$\begin{bmatrix} 0.000 & 0.000 & 1.000 \\ 1.000 & 0.000 & 0.000 \\ 0.000 & 1.000 & 0.000 \end{bmatrix}$

Table C.5: Weights and projection matrices for $H_2^{\ominus}$SE(2) indexed by hypothesis id. Hypothesis H_i created as tuple of SE(2), the projection matrix $T_{\circlearrowleft}$ and its inverse $T_{\circlearrowleft}^{-1} = T_{\circlearrowleft}$.

C.6 Nonholonomic Hypotheses Set $H_3^{\ominus}$SE(2) on SE(2)

Id	Weights	Projection Matrix $T_{\circlearrowleft}$
H_0	[1.0, 1.0]	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 0.924 & -0.383 \\ 0.000 & 0.383 & 0.924 \end{bmatrix}$
H_1	[1.0, 1.0]	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 0.707 & -0.707 \\ 0.000 & 0.707 & 0.707 \end{bmatrix}$
H_2	[1.0, 1.0]	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 0.383 & -0.924 \\ 0.000 & 0.924 & 0.383 \end{bmatrix}$
H_3	[1.0, 1.0]	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 1.000 & 0.000 \\ 0.000 & 0.000 & 1.000 \end{bmatrix}$
H_4	[1.0, 1.0]	$\begin{bmatrix} 0.000 & 0.000 & 1.000 \\ 1.000 & 0.000 & 0.000 \\ 0.000 & 1.000 & 0.000 \end{bmatrix}$

Table C.6: Weights and projection matrices for $H_3^{\ominus}$SE(2) indexed by hypothesis id. Hypothesis H_i created as tuple of SE(2), the projection matrix $T_{\circlearrowleft}$ and its inverse $T_{\circlearrowleft}^{-1} = T_{\circlearrowleft}$.

C.7 Holonomic Hypotheses Set $H^{\oplus}SE(3)$ on $SE(3)$

Id	Weights	Projection Matrix
H_0	I_6	$\begin{bmatrix} 0.000 & 0.000 & 0.707 & 0.000 & 0.000 & -0.707 \\ 0.000 & 0.000 & 0.707 & 0.000 & 0.000 & 0.707 \\ 0.000 & 0.000 & 0.000 & 0.707 & -0.707 & 0.000 \\ 1.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 1.000 & 0.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.000 & 0.707 & 0.707 & 0.000 \end{bmatrix}$
H_1	I_6	$\begin{bmatrix} 0.707 & 0.000 & 0.000 & -0.707 & 0.000 & 0.000 \\ 0.000 & 0.707 & 0.000 & 0.000 & -0.707 & 0.000 \\ 0.000 & 0.000 & 0.707 & 0.000 & 0.000 & -0.707 \\ 0.707 & 0.000 & 0.000 & 0.707 & 0.000 & 0.000 \\ 0.000 & 0.707 & 0.000 & 0.000 & 0.707 & 0.000 \\ 0.000 & 0.000 & 0.707 & 0.000 & 0.000 & 0.707 \end{bmatrix}$
H_2	I_6	$\begin{bmatrix} 0.707 & 0.000 & 0.000 & -0.707 & 0.000 & 0.000 \\ 0.000 & 0.707 & 0.000 & 0.000 & 0.000 & -0.707 \\ 0.000 & 0.000 & 0.707 & 0.000 & -0.707 & 0.000 \\ 0.707 & 0.000 & 0.000 & 0.707 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.707 & 0.000 & 0.707 & 0.000 \\ 0.000 & 0.707 & 0.000 & 0.000 & 0.000 & 0.707 \end{bmatrix}$
H_3	I_6	$\begin{bmatrix} 0.707 & 0.000 & 0.000 & -0.707 & 0.000 & 0.000 \\ 0.707 & 0.000 & 0.000 & 0.707 & 0.000 & 0.000 \\ 0.000 & 0.707 & 0.000 & 0.000 & -0.707 & 0.000 \\ 0.000 & 0.000 & 0.707 & 0.000 & 0.000 & -0.707 \\ 0.000 & 0.707 & 0.000 & 0.000 & 0.707 & 0.000 \\ 0.000 & 0.000 & 0.707 & 0.000 & 0.000 & 0.707 \end{bmatrix}$
H_4	I_6	$\begin{bmatrix} 1.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.707 & 0.000 & -0.707 & 0.000 \\ 0.000 & 0.000 & 0.000 & 0.707 & 0.000 & -0.707 \\ 0.000 & 1.000 & 0.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.707 & 0.000 & 0.707 & 0.000 \\ 0.000 & 0.000 & 0.000 & 0.707 & 0.000 & 0.707 \end{bmatrix}$
H_5	I_6	$\begin{bmatrix} 1.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.707 & 0.000 & 0.000 & -0.707 \\ 0.000 & 0.000 & 0.000 & 0.707 & -0.707 & 0.000 \\ 0.000 & 1.000 & 0.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.000 & 0.707 & 0.707 & 0.000 \\ 0.000 & 0.000 & 0.707 & 0.000 & 0.000 & 0.707 \end{bmatrix}$
H_6	I_6	$\begin{bmatrix} 1.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.707 & 0.000 & 0.000 & -0.707 \\ 0.000 & 0.000 & 0.707 & 0.000 & 0.000 & 0.707 \\ 0.000 & 1.000 & 0.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.000 & 0.707 & -0.707 & 0.000 \\ 0.000 & 0.000 & 0.000 & 0.707 & 0.707 & 0.000 \end{bmatrix}$
H_7	I_6	$\begin{bmatrix} 1.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.707 & 0.000 & -0.707 & 0.000 \\ 0.000 & 0.000 & 0.000 & 0.707 & 0.000 & -0.707 \\ 0.000 & 0.000 & 0.707 & 0.000 & 0.707 & 0.000 \\ 0.000 & 0.000 & 0.000 & 0.707 & 0.000 & 0.707 \\ 0.000 & 1.000 & 0.000 & 0.000 & 0.000 & 0.000 \end{bmatrix}$
H_8	I_6	$\begin{bmatrix} 1.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.707 & 0.000 & 0.000 & -0.707 \\ 0.000 & 0.000 & 0.000 & 0.707 & -0.707 & 0.000 \\ 0.000 & 0.000 & 0.000 & 0.707 & 0.707 & 0.000 \\ 0.000 & 0.000 & 0.707 & 0.000 & 0.000 & 0.707 \\ 0.000 & 1.000 & 0.000 & 0.000 & 0.000 & 0.000 \end{bmatrix}$
H_9	I_6	$\begin{bmatrix} 1.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 1.000 & 0.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.000 & 0.000 & 0.707 & -0.707 \\ 0.000 & 0.000 & 1.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.000 & 1.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.000 & 0.000 & 0.707 & 0.707 \end{bmatrix}$
H_{10}	I_6	$\begin{bmatrix} 1.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 1.000 & 0.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.000 & 0.000 & 0.707 & -0.707 \\ 0.000 & 0.000 & 0.000 & 0.000 & 0.707 & 0.707 \\ 0.000 & 0.000 & 1.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.000 & 1.000 & 0.000 & 0.000 \end{bmatrix}$
H_{11}	I_6	$\begin{bmatrix} 1.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 1.000 & 0.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.707 & 0.000 & 0.000 & -0.707 \\ 0.000 & 0.000 & 0.707 & 0.000 & 0.000 & 0.707 \\ 0.000 & 0.000 & 0.000 & 0.707 & -0.707 & 0.000 \\ 0.000 & 0.000 & 0.000 & 0.707 & 0.707 & 0.000 \end{bmatrix}$
H_{12}	I_6	$\begin{bmatrix} 1.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 1.000 & 0.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.707 & 0.000 & -0.707 & 0.000 \\ 0.000 & 0.000 & 0.000 & 0.707 & 0.000 & -0.707 \\ 0.000 & 0.000 & 0.707 & 0.000 & 0.707 & 0.000 \\ 0.000 & 0.000 & 0.000 & 0.707 & 0.000 & 0.707 \end{bmatrix}$
H_{13}	I_6	$\begin{bmatrix} 1.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 1.000 & 0.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & 1.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.000 & 1.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.000 & 0.000 & 0.707 & -0.707 \\ 0.000 & 0.000 & 0.000 & 0.000 & 0.707 & 0.707 \end{bmatrix}$
H_{14}	I_6	$\begin{bmatrix} 1.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 1.000 & 0.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & 1.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.000 & 1.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.000 & 0.000 & 1.000 & 0.000 \\ 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 1.000 \end{bmatrix}$

Table C.7: Weights and projection matrices for $H^{\oplus}SE(3)$ indexed by hypothesis id. In this table I_6 abbreviates the weight vector $[1.0, 1.0, 1.0, 1.0, 1.0, 1.0]$. Hypothesis H_i created as tuple of SE(3), the projection matrix $T_{\cup}$ and its inverse $T_{\cup}^{-1} = T_{\cup}$.

C.8 Nonholonomic Hypotheses Set $H^{\ominus}SE(3)$ on $SE(3)$

Id	Weights	Projection Matrix
H_0	[1.0, 1.0]	$\begin{bmatrix} 0.000 & 0.000 & 1.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.000 & 1.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.000 & 0.000 & 1.000 & 0.000 \\ 1.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 1.000 & 0.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 1.000 \end{bmatrix}$
H_1	[1.0, 1.0]	$\begin{bmatrix} 0.000 & 0.707 & 0.000 & 0.000 & 0.000 & -0.707 \\ 0.000 & 0.000 & 1.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.000 & 1.000 & 0.000 & 0.000 \\ 1.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.707 & 0.000 & 0.000 & 0.000 & 0.707 \\ 0.000 & 0.000 & 0.000 & 0.000 & 1.000 & 0.000 \end{bmatrix}$
H_2	[1.0, 1.0]	$\begin{bmatrix} 0.707 & 0.000 & 0.000 & 0.000 & 0.000 & -0.707 \\ 0.000 & 0.000 & 1.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.000 & 1.000 & 0.000 & 0.000 \\ 0.707 & 0.000 & 0.000 & 0.000 & 0.000 & 0.707 \\ 0.000 & 0.707 & 0.000 & 0.000 & -0.707 & 0.000 \\ 0.000 & 0.707 & 0.000 & 0.000 & 0.707 & 0.000 \end{bmatrix}$
H_3	[1.0, 1.0]	$\begin{bmatrix} 0.707 & 0.000 & 0.000 & 0.000 & 0.000 & -0.707 \\ 0.707 & 0.000 & 0.000 & 0.000 & 0.000 & 0.707 \\ 0.000 & 0.000 & 1.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.707 & 0.000 & 0.000 & -0.707 & 0.000 \\ 0.000 & 0.707 & 0.000 & 0.000 & 0.707 & 0.000 \\ 0.000 & 0.000 & 0.000 & 1.000 & 0.000 & 0.000 \end{bmatrix}$

Id	Weights	Projection Matrix
H_4	[1.0, 1.0]	$\begin{bmatrix} 0.707 & 0.000 & 0.000 & 0.000 & -0.707 & 0.000 \\ 0.707 & 0.000 & 0.000 & 0.000 & 0.707 & 0.000 \\ 0.000 & 0.707 & 0.000 & 0.000 & 0.000 & -0.707 \\ 0.000 & 0.000 & 1.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.707 & 0.000 & 0.000 & 0.000 & 0.707 \\ 0.000 & 0.000 & 0.000 & 1.000 & 0.000 & 0.000 \end{bmatrix}$
H_5	[1.0, 1.0]	$\begin{bmatrix} 1.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & 1.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.000 & 1.000 & 0.000 & 0.000 \\ 0.000 & 1.000 & 0.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.000 & 0.000 & 1.000 & 0.000 \\ 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 1.000 \end{bmatrix}$
H_6	[1.0, 1.0]	$\begin{bmatrix} 1.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.707 & 0.000 & 0.000 & 0.000 & -0.707 \\ 0.000 & 0.000 & 1.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.707 & 0.000 & 0.000 & 0.000 & 0.707 \\ 0.000 & 0.000 & 0.000 & 1.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.000 & 0.000 & 1.000 & 0.000 \end{bmatrix}$
H_7	[1.0, 1.0]	$\begin{bmatrix} 1.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 1.000 & 0.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & 1.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.000 & 1.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.000 & 0.000 & 1.000 & 0.000 \\ 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 1.000 \end{bmatrix}$

Table C.8: Weights and projection matrices for $H^{\ominus}SE(3)$ indexed by hypothesis id. Hypothesis H_i created as tuple of SE(3), the projection matrix $T_{\circlearrowleft}$ and its inverse $T_{\circlearrowleft}^{-1} = T_{\circlearrowleft}$.

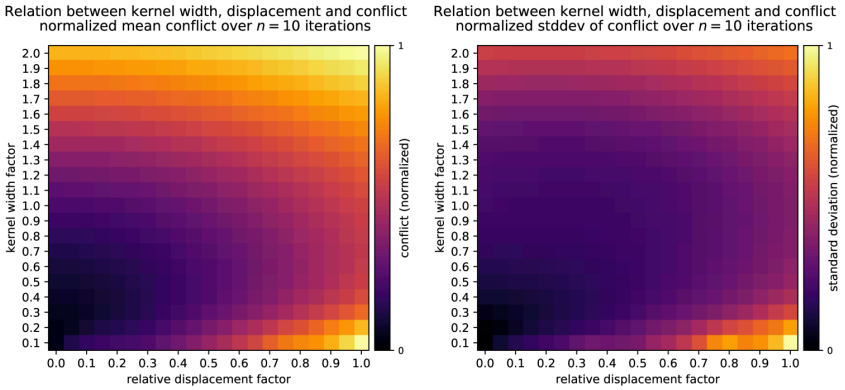
D

Additional Tables and Figures

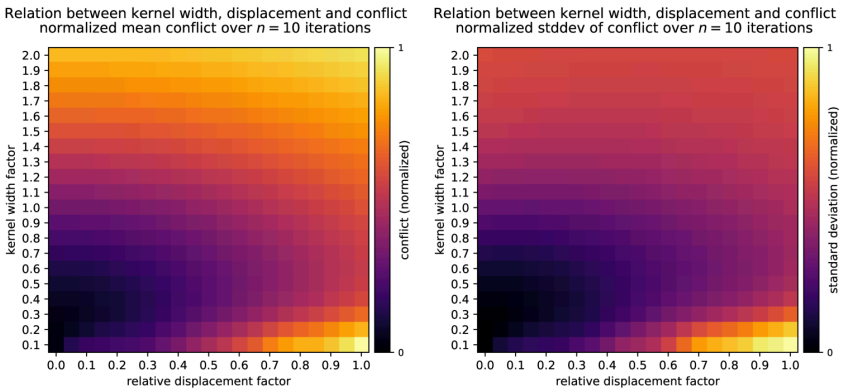
This appendix contains additional tables and figures, visualizing the results of the experiments conducted in Chapter 7. For an overview on the figures and the notation we refer to Chapter 7.

D.1 Results from Scenario I : Conflict Function Evaluation

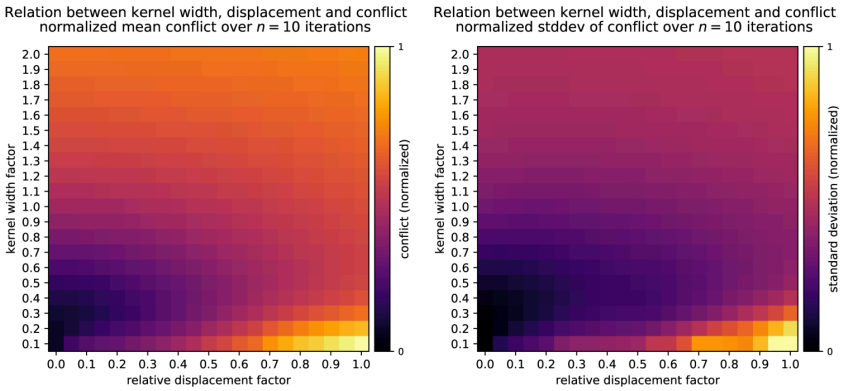
D.1.1 Conflict Function Evaluation of Δ_{NW}



(a) Mean and standard deviation of the Δ_{NW} conflict function for a step size of 0.5.



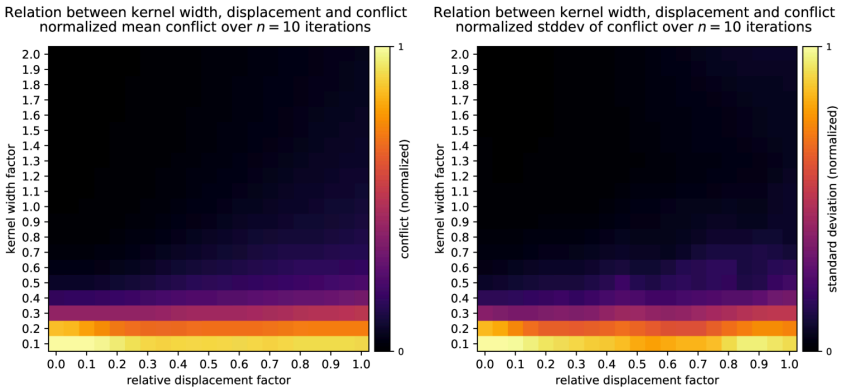
(b) Mean and standard deviation of the Δ_{NW} conflict function for a step size of 1.0.



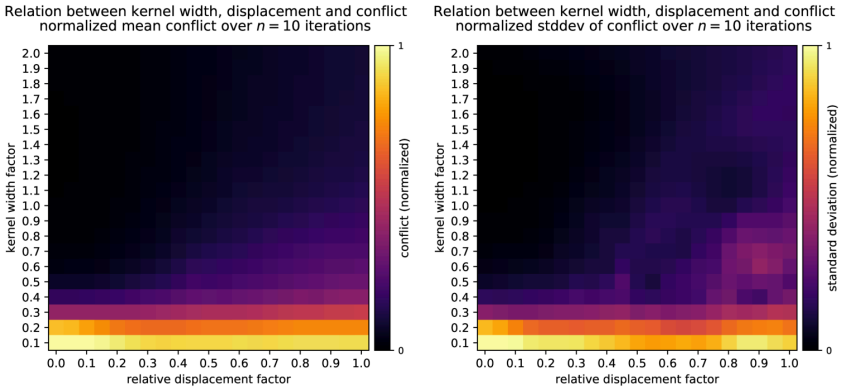
(c) Mean and standard deviation of the Δ_{NW} conflict function for a step size of 1.5.

Figure D.1: Mean and standard deviation for the Δ_{NW} conflict function and various step sizes.

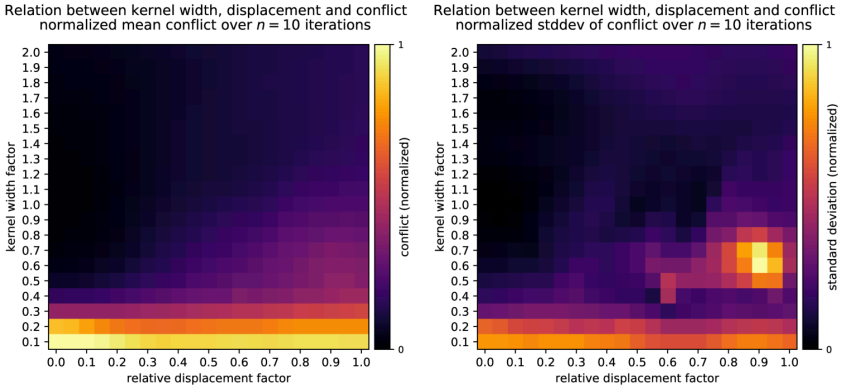
D.1.2 Conflict Function Evaluation of Δ_{RBFN}



(a) Mean and standard deviation of the Δ_{RBFN} conflict function for a step size of 0.5.



(b) Mean and standard deviation of the Δ_{RBFN} conflict function for a step size of 1.0.

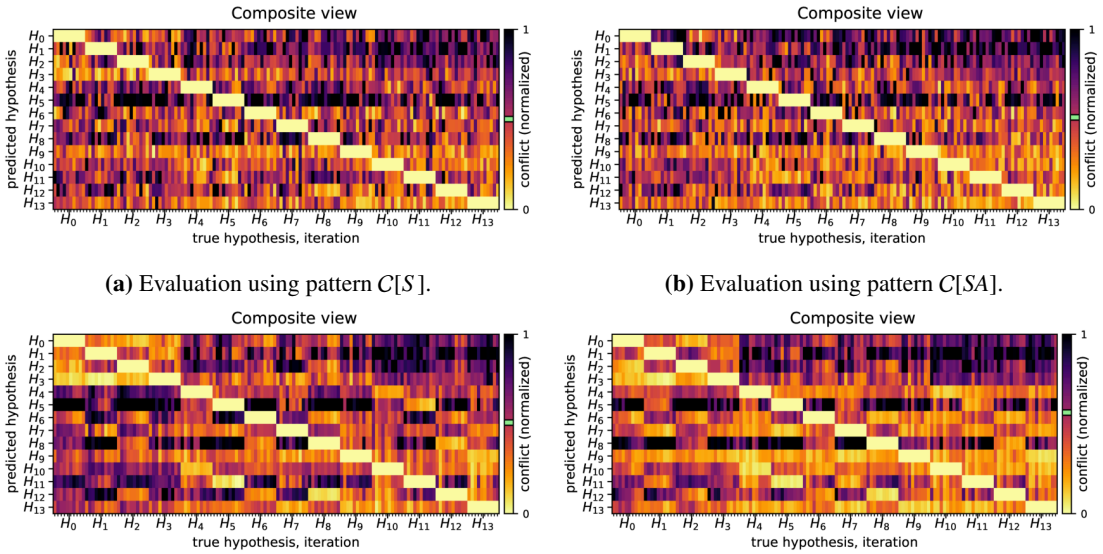


(c) Mean and standard deviation of the Δ_{RBFN} conflict function for a step size of 1.5.

Figure D.2: Mean and standard deviation for the Δ_{RBFN} conflict function and various step sizes.

D.2 Scenario IIa : Holonomic Pattern Evaluation on SE(2)

D.2.1 Pattern and Step size Evaluation on $H_1^{\oplus}SE(2)$ and $H_2^{\oplus}SE(2)$ utilizing Δ_{NW}



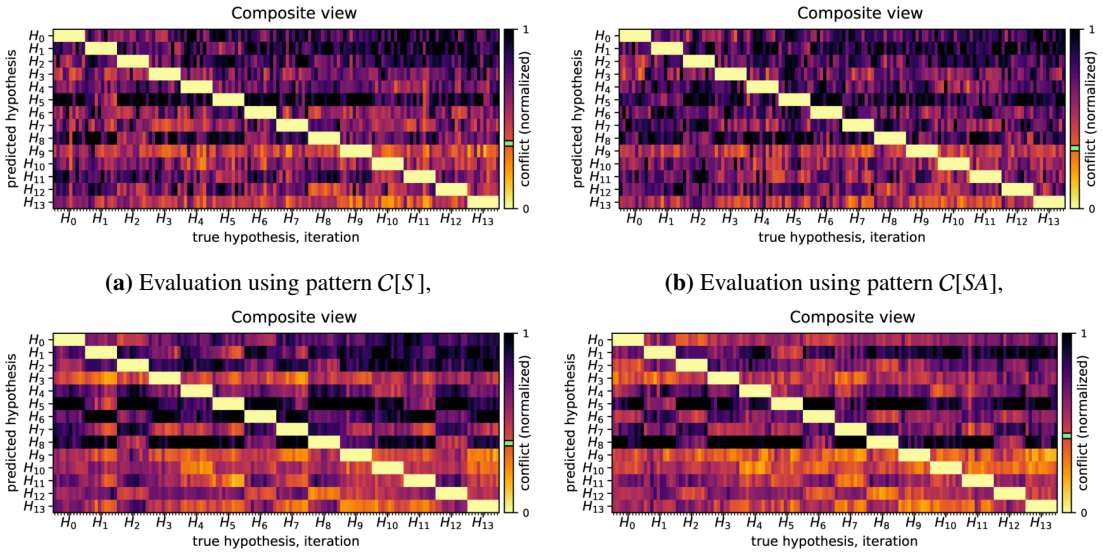
(a) Evaluation using pattern $C[S]$.

(b) Evaluation using pattern $C[SA]$.

(c) Evaluation using pattern $C[R]$.

(d) Evaluation using pattern $C[RA]$.

Figure D.3: Pattern and step size evaluation on $H_1^{\oplus}SE(2)$ utilizing Δ_{NW} with a step size of 0.5.



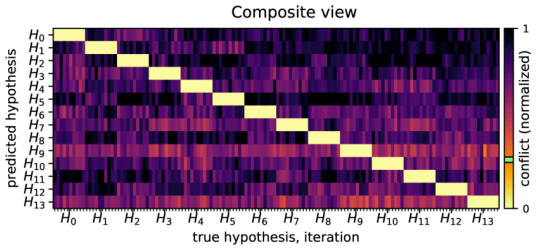
(a) Evaluation using pattern $C[S]$,

(b) Evaluation using pattern $C[SA]$,

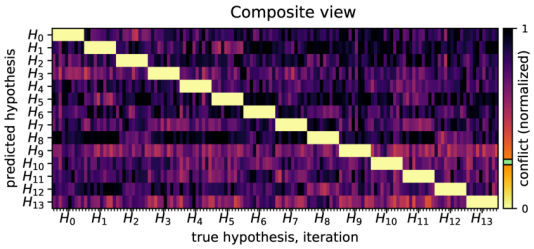
(c) Evaluation using pattern $C[R]$,

(d) Evaluation using pattern $C[RA]$,

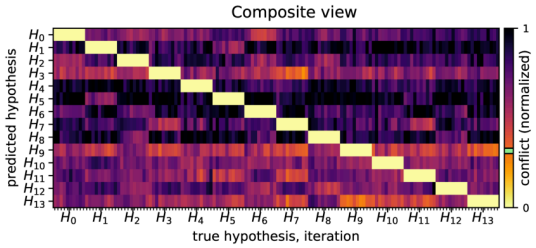
Figure D.4: Pattern and step size evaluation on $H_1^{\oplus}SE(2)$ utilizing Δ_{NW} with a step size of 1.0.



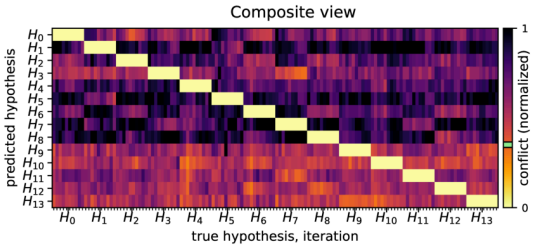
(a) Evaluation using pattern $C[S]$.



(b) Evaluation using pattern $C[SA]$.

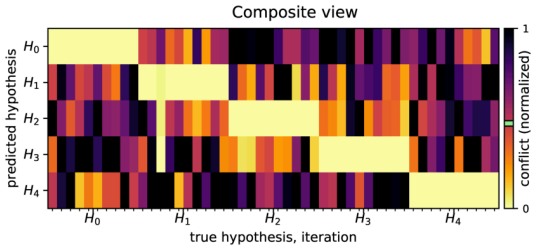


(c) Evaluation using pattern $C[R]$.

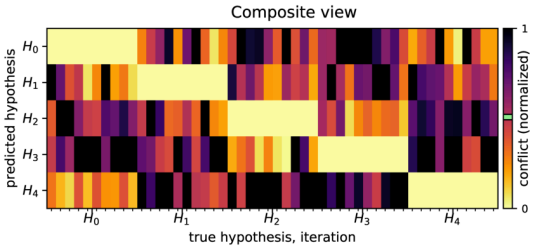


(d) Evaluation using pattern $C[RA]$.

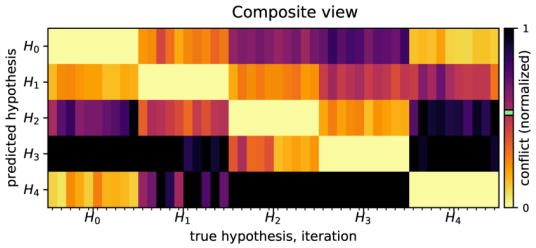
Figure D.5: Pattern and step size evaluation on $\mathrm{H}_1^{\oplus}\mathrm{SE}(2)$ utilizing Δ_{NW} with a step size of 1.5.



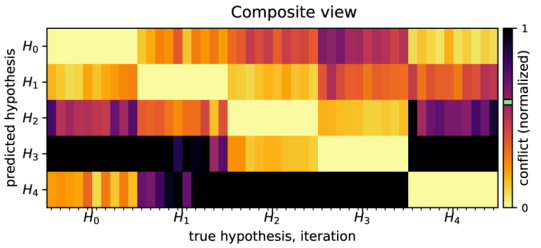
(a) Evaluation using pattern $C[S]$.



(b) Evaluation using pattern $C[SA]$.

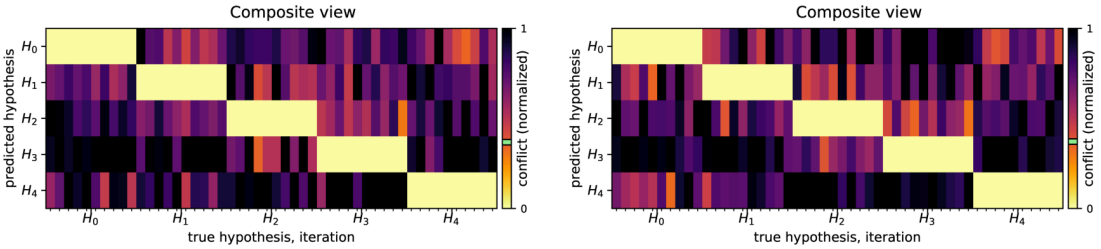


(c) Evaluation using pattern $C[R]$.



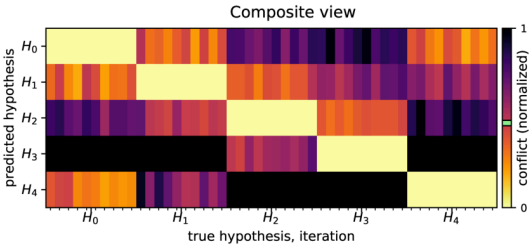
(d) Evaluation using pattern $C[RA]$.

Figure D.6: Pattern and step size evaluation on $\mathrm{H}_2^{\oplus}\mathrm{SE}(2)$ utilizing Δ_{NW} with a step size of 0.5.

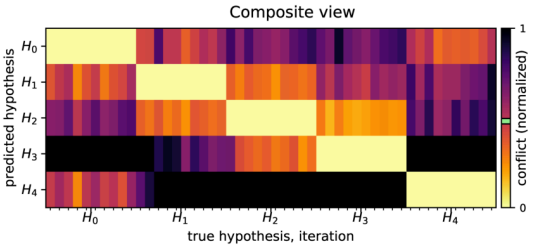


(a) Evaluation using pattern $C[S]$.

(b) Evaluation using pattern $C[SA]$.

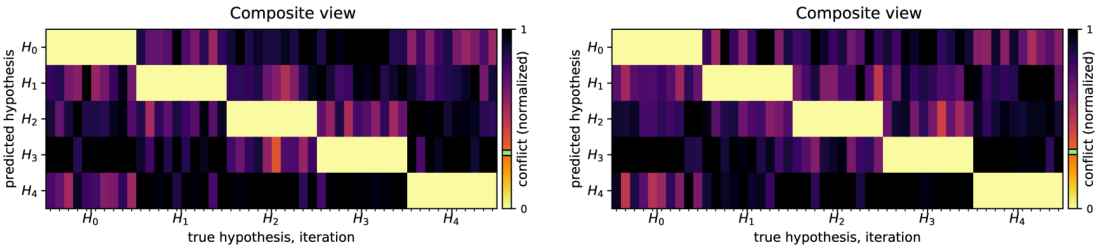


(c) Evaluation using pattern $C[R]$.



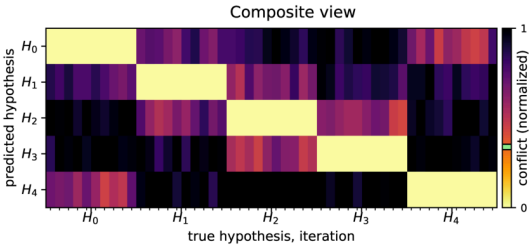
(d) Evaluation using pattern $C[RA]$.

Figure D.7: Pattern and step size evaluation on $\mathrm{H}_2^{\oplus}\mathrm{SE}(2)$ utilizing Δ_{NW} with a step size of 1.0.

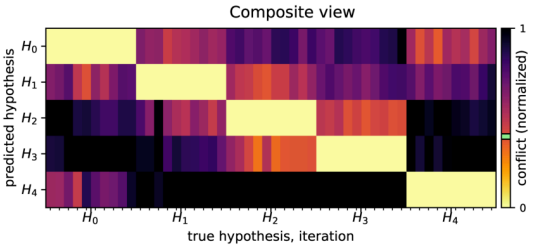


(a) Evaluation using pattern $C[S]$.

(b) Evaluation using pattern $C[SA]$.



(c) Evaluation using pattern $C[R]$.



(d) Evaluation using pattern $C[RA]$.

Figure D.8: Pattern and step size evaluation on $\mathrm{H}_2^{\oplus}\mathrm{SE}(2)$ and utilizing Δ_{NW} with a step size of 1.5.

Stepsize	Pattern	$\Delta[\Xi]/\sum I$		Accuracy
-	-	μ	σ	-
0.5	$C[S]$	0.502	0.283	1.00
0.5	$C[SA]$	0.511	0.284	1.00
0.5	$C[R]$	0.508	0.288	1.00
0.5	$C[RA]$	0.563	0.287	1.00
1.0	$C[S]$	0.363	0.256	1.00
1.0	$C[SA]$	0.337	0.254	1.00
1.0	$C[R]$	0.387	0.266	1.00
1.0	$C[RA]$	0.428	0.254	1.00
1.5	$C[S]$	0.271	0.242	1.00
1.5	$C[SA]$	0.258	0.243	1.00
1.5	$C[R]$	0.316	0.252	1.00
1.5	$C[RA]$	0.351	0.249	1.00

(a) Pattern and step size evaluation on $H_1^{\oplus}SE(2)$.

Stepsize	Pattern	$\Delta[\Xi]/\sum I$		Accuracy
-	-	μ	σ	-
0.5	$C[S]$	0.472	0.373	0.98
0.5	$C[SA]$	0.508	0.367	1.00
0.5	$C[R]$	0.530	0.374	1.00
0.5	$C[RA]$	0.588	0.369	1.00
1.0	$C[S]$	0.368	0.356	1.00
1.0	$C[SA]$	0.372	0.359	1.00
1.0	$C[R]$	0.472	0.355	1.00
1.0	$C[RA]$	0.482	0.351	1.00
1.5	$C[S]$	0.311	0.366	1.00
1.5	$C[SA]$	0.316	0.363	1.00
1.5	$C[R]$	0.338	0.365	1.00
1.5	$C[RA]$	0.397	0.355	1.00

(b) Pattern and step size evaluation on $H_2^{\oplus}SE(2)$.

Table D.1: Pattern and step size evaluation on $H_1^{\oplus}SE(2)$ and $H_2^{\oplus}SE(2)$ utilizing Δ_{NW} with various patterns and step sizes. Intrinsic conflict (μ and σ) and classification accuracy.

Stepsize	Pattern	$\Lambda[\Xi]$		$\mathcal{S}$		$\mathcal{B}$		T_{CPU}		T_{WALL}		$\mathcal{S}/\Lambda[\Xi]$		$\mathcal{B}/\Lambda[\Xi]$	
-	-	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
0.5	$C[S]$	196.9	6.1	1802.8	16.1	1863.4	26.3	110.431	2.419	1.748	0.037	9.2	0.3	9.5	0.4
0.5	$C[SA]$	188.5	7.4	1834.1	22.7	1874.3	19.6	105.106	2.384	1.666	0.043	9.7	0.5	10.0	0.5
0.5	$C[R]$	137.1	9.2	4125.7	74.1	4130.5	73.4	259.547	7.755	4.644	0.164	30.2	2.5	30.3	2.4
0.5	$C[RA]$	128.0	3.9	4116.4	62.0	4116.8	62.0	251.221	7.930	4.584	0.138	32.2	1.3	32.2	1.3
1.0	$C[S]$	168.4	2.8	1900.5	14.4	2075.3	25.9	119.852	3.267	2.091	0.067	11.3	0.2	12.3	0.3
1.0	$C[SA]$	161.1	7.6	1900.2	21.7	2038.7	24.7	114.838	2.499	2.020	0.069	11.8	0.7	12.7	0.7
1.0	$C[R]$	58.0	2.3	2152.8	34.2	2171.0	31.7	159.800	5.230	2.886	0.080	37.2	1.9	37.5	1.9
1.0	$C[RA]$	105.3	2.2	4280.1	41.0	4300.1	42.6	305.566	12.787	5.560	0.200	40.7	1.0	40.9	1.1
1.5	$C[S]$	142.4	4.1	2021.9	11.4	2304.8	19.3	134.219	2.804	2.592	0.071	14.2	0.5	16.2	0.6
1.5	$C[SA]$	138.0	5.7	1994.9	21.8	2236.8	22.6	129.995	2.198	2.534	0.080	14.5	0.8	16.2	0.8
1.5	$C[R]$	14.2	0.6	640.7	3.2	657.6	2.8	65.608	1.969	1.146	0.030	45.2	2.0	46.3	1.9
1.5	$C[RA]$	17.8	0.4	897.7	6.4	915.5	6.4	92.657	4.112	1.611	0.066	50.4	1.5	51.4	1.4

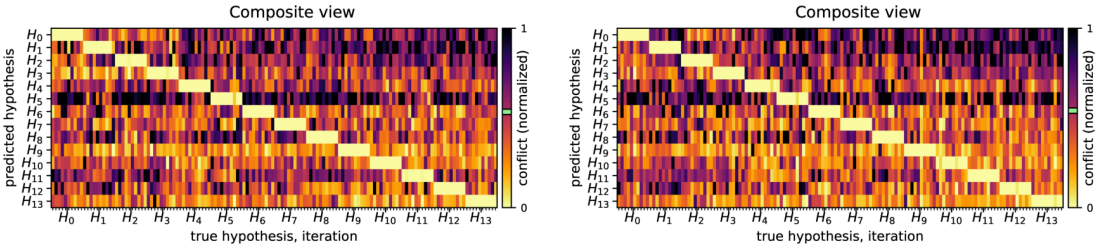
(a) Pattern and step size evaluation on $H_1^{\oplus}SE(2)$.

Stepsize	Pattern	$\Lambda[\Xi]$		$\mathcal{S}$		$\mathcal{B}$		T_{CPU}		T_{WALL}		$\mathcal{S}/\Lambda[\Xi]$		$\mathcal{B}/\Lambda[\Xi]$	
-	-	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
0.5	$C[S]$	186.8	11.2	1825.0	27.9	1879.6	47.1	107.046	4.582	1.723	0.092	9.8	0.6	10.1	0.8
0.5	$C[SA]$	172.9	10.5	1851.4	30.4	1890.6	22.6	100.979	4.379	1.628	0.066	10.7	0.8	11.0	0.7
0.5	$C[R]$	125.2	8.4	4230.4	81.8	4234.0	80.6	276.141	14.561	4.904	0.226	34.0	2.6	34.0	2.6
0.5	$C[RA]$	119.9	4.2	4097.9	54.2	4098.2	54.1	250.444	6.086	4.590	0.088	34.2	1.3	34.2	1.3
1.0	$C[S]$	144.9	4.2	1924.9	23.0	2090.8	51.9	112.751	6.404	2.083	0.150	13.3	0.5	14.4	0.7
1.0	$C[SA]$	135.9	9.6	1938.8	29.6	2084.6	33.0	110.509	2.735	2.112	0.091	14.3	1.3	15.4	1.3
1.0	$C[R]$	19.0	1.1	851.0	14.9	855.1	13.9	61.426	2.477	1.098	0.026	45.0	3.2	45.2	3.1
1.0	$C[RA]$	46.7	1.1	2144.8	25.5	2153.2	25.8	152.231	5.719	2.732	0.062	46.0	1.2	46.1	1.2
1.5	$C[S]$	111.4	6.3	2042.3	20.8	2301.2	49.6	133.218	11.275	2.738	0.229	18.4	1.2	20.7	1.5
1.5	$C[SA]$	105.3	5.9	2033.3	36.7	2305.4	30.8	140.264	6.539	2.883	0.124	19.4	1.4	22.0	1.4
1.5	$C[R]$	8.6	0.4	490.5	9.9	500.7	8.6	50.994	2.403	0.858	0.035	57.1	3.5	58.3	3.4
1.5	$C[RA]$	10.5	0.4	632.0	10.4	641.8	11.3	64.872	4.241	1.092	0.065	60.1	2.4	61.0	2.4

(b) Pattern and step size evaluation on $H_2^{\oplus}SE(2)$.

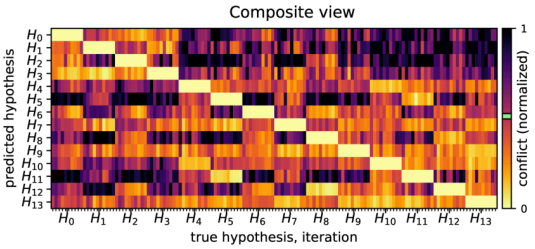
Table D.2: Pattern and step size evaluation on $H_1^{\oplus}SE(2)$ and $H_2^{\oplus}SE(2)$ utilizing Δ_{NW} with various patterns and step sizes. Sensorimotor map properties and timing data.

D.2.2 Pattern and step size evaluation on $H_1^{\oplus}SE(2)$ and $H_2^{\oplus}SE(2)$ utilizing Δ_{RBFN}

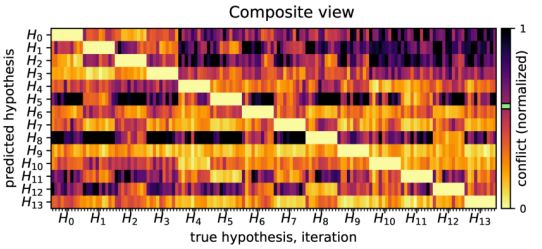


(a) Evaluation using pattern $C[S]$.

(b) Evaluation using pattern $C[SA]$.

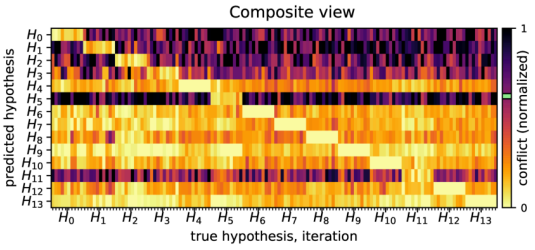


(c) Evaluation using pattern $C[R]$.

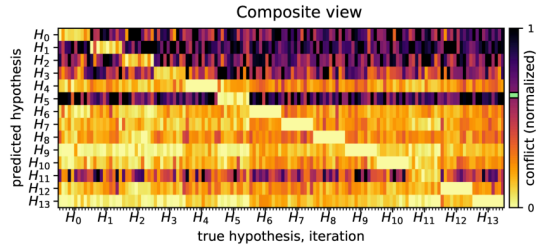


(d) Evaluation using pattern $C[RA]$.

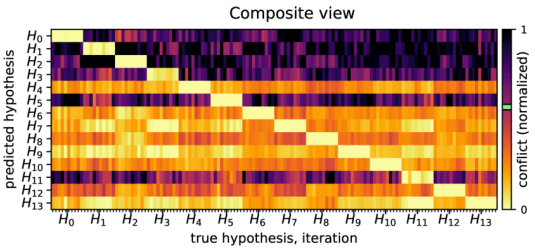
Figure D.9: Pattern and step size evaluation on $H_1^{\oplus}SE(2)$ utilizing Δ_{RBFN} with a step size of 0.5.



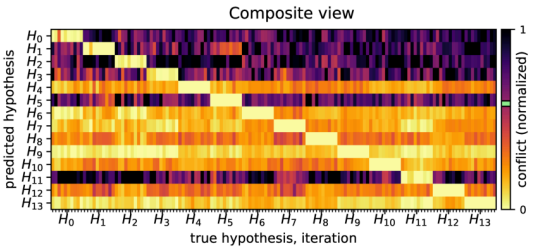
(a) Evaluation using pattern $C[S]$.



(b) Evaluation using pattern $C[SA]$.

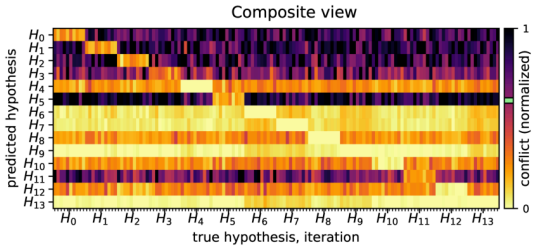


(c) Evaluation using pattern $C[R]$.

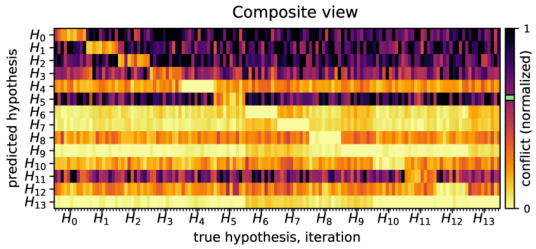


(d) Evaluation using pattern $C[RA]$.

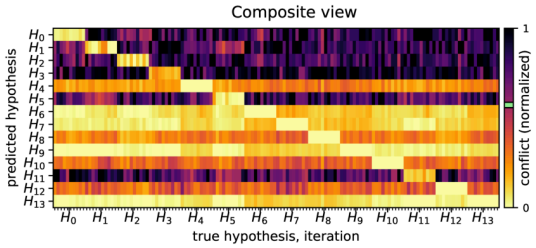
Figure D.10: Pattern and step size evaluation on $H_1^{\oplus}SE(2)$ utilizing Δ_{RBFN} with a step size of 1.0.



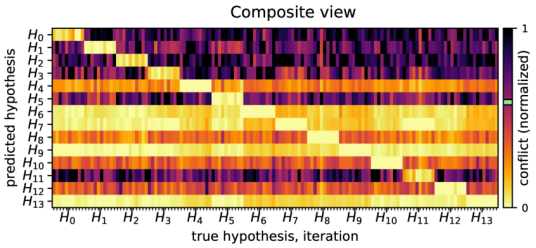
(a) Evaluation using pattern $C[S]$.



(b) Evaluation using pattern $C[SA]$.

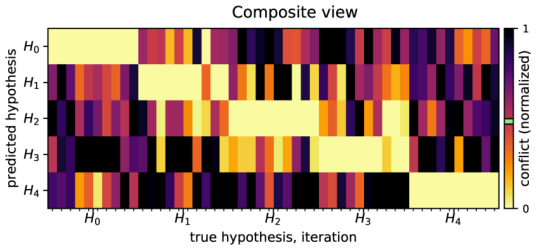


(c) Evaluation using pattern $C[R]$.

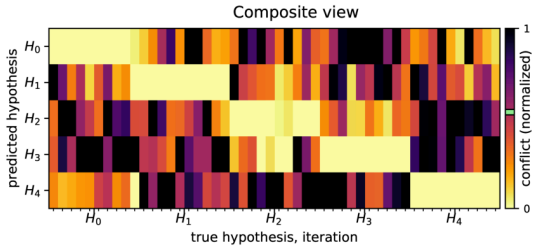


(d) Evaluation using pattern $C[RA]$.

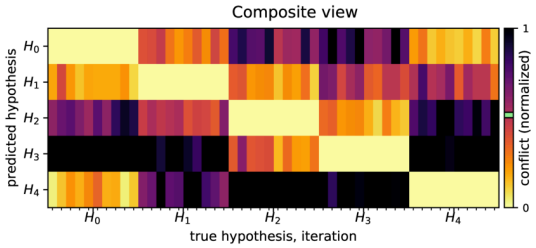
Figure D.11: Pattern and step size evaluation on $\mathrm{H}_1^{\oplus}\mathrm{SE}(2)$ utilizing Δ_{RBFN} with a step size of 1.5.



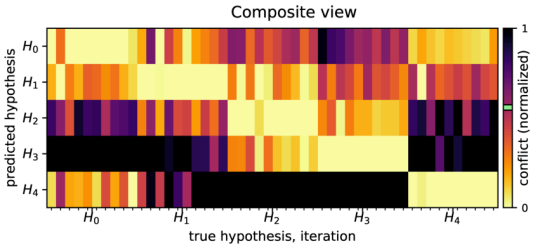
(a) Evaluation using pattern $C[S]$.



(b) Evaluation using pattern $C[SA]$.

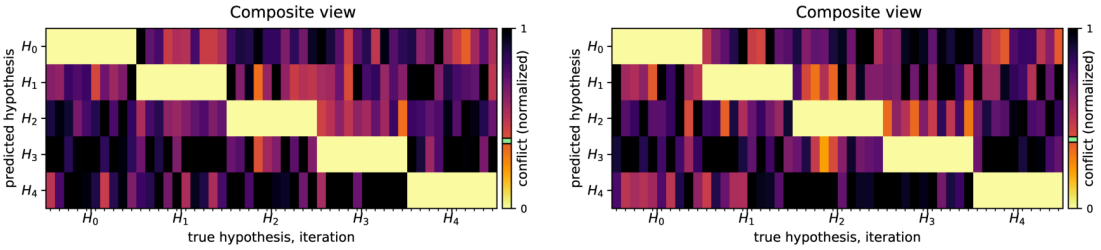


(c) Evaluation using pattern $C[R]$.



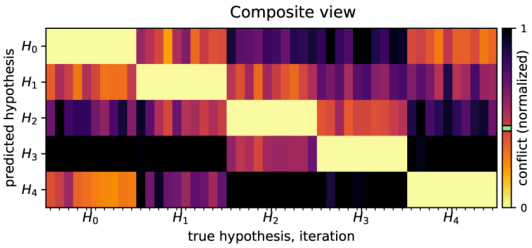
(d) Evaluation using pattern $C[RA]$.

Figure D.12: Pattern and step size evaluation on $\mathrm{H}_2^{\oplus}\mathrm{SE}(2)$ utilizing Δ_{RBFN} with a step size of 0.5.

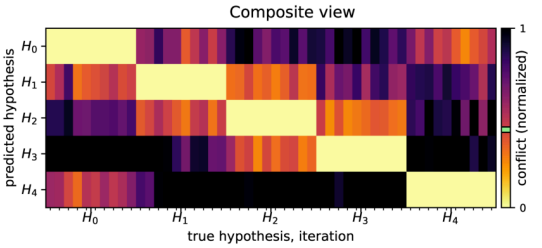


(a) Evaluation using pattern $C[S]$.

(b) Evaluation using pattern $C[SA]$.

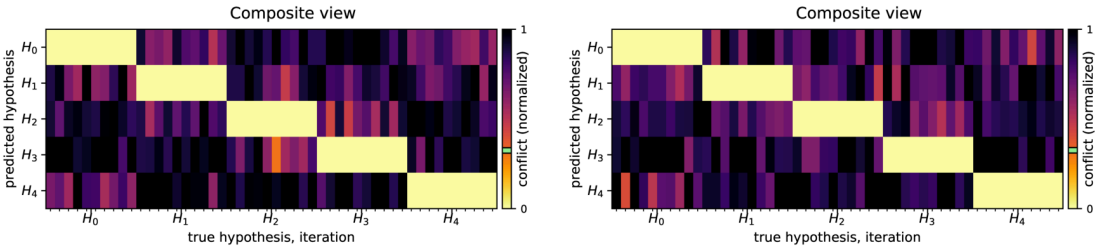


(c) Evaluation using pattern $C[R]$.



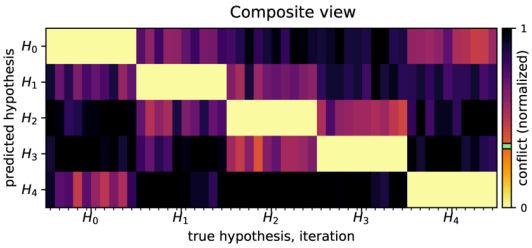
(d) $C[RA]$

Figure D.13: Pattern and step size evaluation on $\mathrm{H}_2^{\oplus}\mathrm{SE}(2)$ utilizing Δ_{RBFN} with a step size of 1.0.

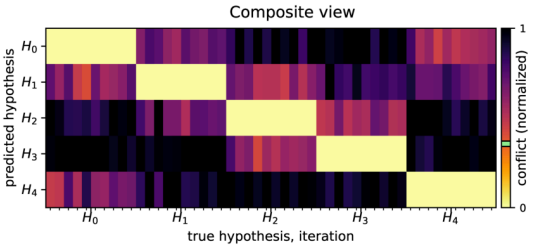


(a) Evaluation using pattern $C[S]$.

(b) Evaluation using pattern $C[SA]$.



(c) Evaluation using pattern $C[R]$.



(d) Evaluation using pattern $C[RA]$.

Figure D.14: Pattern and step size evaluation on $\mathrm{H}_2^{\oplus}\mathrm{SE}(2)$ utilizing Δ_{RBFN} and stepsize of 1.5

Stepsize	Pattern	$\Delta[\Xi]/\sum I$		Accuracy
-	-	μ	σ	-
0.5	$C[S]$	0.535	0.284	0.95
0.5	$C[SA]$	0.541	0.289	0.95
0.5	$C[R]$	0.512	0.295	1.00
0.5	$C[RA]$	0.568	0.293	1.00
1.0	$C[S]$	0.621	0.320	0.61
1.0	$C[SA]$	0.627	0.314	0.64
1.0	$C[R]$	0.569	0.328	0.86
1.0	$C[RA]$	0.588	0.327	0.81
1.5	$C[S]$	0.600	0.344	0.45
1.5	$C[SA]$	0.615	0.333	0.44
1.5	$C[R]$	0.573	0.347	0.60
1.5	$C[RA]$	0.588	0.330	0.62

(a) Pattern and step size evaluation on $H_1^{\oplus}SE(2)$.

Stepsize	Pattern	$\Delta[\Xi]/\sum I$		Accuracy
-	-	μ	σ	-
0.5	$C[S]$	0.483	0.380	0.94
0.5	$C[SA]$	0.535	0.375	0.92
0.5	$C[R]$	0.517	0.376	1.00
0.5	$C[RA]$	0.559	0.378	0.90
1.0	$C[S]$	0.375	0.355	1.00
1.0	$C[SA]$	0.383	0.359	1.00
1.0	$C[R]$	0.442	0.356	1.00
1.0	$C[RA]$	0.435	0.363	1.00
1.5	$C[S]$	0.324	0.365	1.00
1.5	$C[SA]$	0.325	0.361	1.00
1.5	$C[R]$	0.343	0.362	1.00
1.5	$C[RA]$	0.356	0.360	1.00

(b) Pattern and step size evaluation on $H_2^{\oplus}SE(2)$.

Table D.3: Pattern and step size evaluation on $H_1^{\oplus}SE(2)$ and $H_2^{\oplus}SE(2)$ utilizing Δ_{RBFN} with various patterns and step sizes. Intrinsic conflict (μ and σ) and classification accuracy.

Stepsize	Pattern	$\Lambda[\Xi]$		$\mathcal{S}$		$\mathcal{B}$		T_{CPU}		T_{WALL}		$\mathcal{S}/\Lambda[\Xi]$		$\mathcal{B}/\Lambda[\Xi]$	
-	-	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
0.5	$C[S]$	196.9	6.1	1802.8	16.1	1863.4	26.3	125.521	3.078	1.783	0.043	9.2	0.3	9.5	0.4
0.5	$C[SA]$	188.5	7.4	1834.1	22.7	1874.3	19.6	119.760	2.855	1.693	0.043	9.7	0.5	10.0	0.5
0.5	$C[R]$	137.1	9.2	4125.7	74.1	4130.5	73.4	224.885	3.545	4.436	0.158	30.2	2.5	30.3	2.4
0.5	$C[RA]$	128.0	3.9	4116.4	62.0	4116.8	62.0	213.238	3.707	4.351	0.150	32.2	1.3	32.2	1.3
1.0	$C[S]$	168.4	2.8	1900.5	14.4	2075.3	25.9	133.489	1.827	2.006	0.039	11.3	0.2	12.3	0.3
1.0	$C[SA]$	161.1	7.6	1900.2	21.7	2038.7	24.7	129.460	3.128	1.948	0.050	11.8	0.7	12.7	0.7
1.0	$C[R]$	58.0	2.3	2152.8	34.2	2171.0	31.7	125.108	3.037	2.727	0.096	37.2	1.9	37.5	1.9
1.0	$C[RA]$	105.3	2.2	4280.1	41.0	4300.1	42.6	237.023	7.861	5.249	0.213	40.7	1.0	40.9	1.1
1.5	$C[S]$	142.4	4.1	2021.9	11.4	2304.8	19.3	141.063	2.335	2.330	0.055	14.2	0.5	16.2	0.6
1.5	$C[SA]$	138.0	5.7	1994.9	21.8	2236.8	22.6	139.191	1.472	2.309	0.059	14.5	0.8	16.2	0.8
1.5	$C[R]$	14.2	0.6	640.7	3.2	657.6	2.8	47.237	1.180	1.099	0.033	45.2	2.0	46.3	1.9
1.5	$C[RA]$	17.8	0.4	897.7	6.4	915.5	6.4	64.859	2.598	1.536	0.065	50.4	1.5	51.4	1.4

(a) Pattern and step size evaluation on $H_1^{\oplus}SE(2)$.

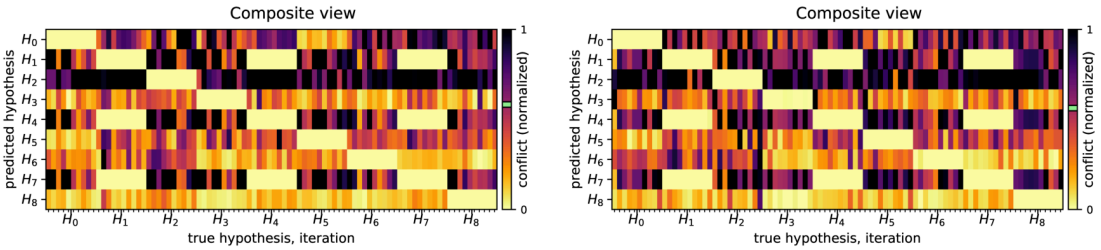
Stepsize	Pattern	$\Lambda[\Xi]$		$\mathcal{S}$		$\mathcal{B}$		T_{CPU}		T_{WALL}		$\mathcal{S}/\Lambda[\Xi]$		$\mathcal{B}/\Lambda[\Xi]$	
-	-	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
0.5	$C[S]$	186.8	11.2	1825.0	27.9	1879.6	47.1	122.222	3.599	1.755	0.076	9.8	0.6	10.1	0.8
0.5	$C[SA]$	172.9	10.5	1851.4	30.4	1890.6	22.6	114.978	4.156	1.632	0.066	10.7	0.8	11.0	0.7
0.5	$C[R]$	125.2	8.4	4230.4	81.8	4234.0	80.6	233.746	6.382	4.827	0.214	34.0	2.6	34.0	2.6
0.5	$C[RA]$	119.9	4.2	4097.9	54.2	4098.2	54.1	212.731	4.372	4.414	0.118	34.2	1.3	34.2	1.3
1.0	$C[S]$	144.9	4.2	1924.9	23.0	2090.8	51.9	126.111	5.258	1.993	0.101	13.3	0.5	14.4	0.7
1.0	$C[SA]$	135.9	9.6	1938.8	29.6	2084.6	33.0	120.828	4.025	1.972	0.054	14.3	1.3	15.4	1.3
1.0	$C[R]$	19.0	1.1	851.0	14.9	855.1	13.9	46.600	1.465	1.061	0.042	45.0	3.2	45.2	3.1
1.0	$C[RA]$	46.7	1.1	2144.8	25.5	2153.2	25.8	116.359	3.445	2.618	0.086	46.0	1.2	46.1	1.2
1.5	$C[S]$	111.4	6.3	2042.3	20.8	2301.2	49.6	128.560	5.235	2.344	0.163	18.4	1.2	20.7	1.5
1.5	$C[SA]$	105.3	5.9	2033.3	36.7	2305.4	30.8	130.366	4.205	2.448	0.115	19.4	1.4	22.0	1.4
1.5	$C[R]$	8.6	0.4	490.5	9.9	500.7	8.6	35.581	1.288	0.841	0.037	57.1	3.5	58.3	3.4
1.5	$C[RA]$	10.5	0.4	632.0	10.4	641.8	11.3	44.789	2.789	1.067	0.069	60.1	2.4	61.0	2.4

(b) Pattern and step size evaluation on $H_2^{\oplus}SE(2)$.

Table D.4: Pattern and step size evaluation on $H_1^{\oplus}SE(2)$ and $H_2^{\oplus}SE(2)$ utilizing Δ_{RBFN} with various patterns and step sizes. Sensorimotor map properties and timing data.

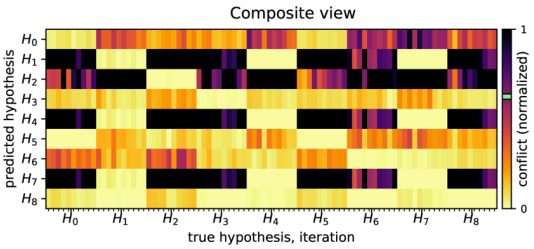
D.3 Scenario IIb : Nonholonomic Pattern Evaluation on SE(2)

D.3.1 Pattern and Step size Evaluation on $H_1^{\ominus}SE(2)$ and $H_2^{\ominus}SE(2)$ utilizing Δ_{NW}

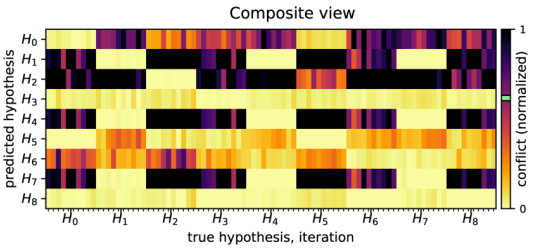


(a) Evaluation using pattern $C[S]$.

(b) Evaluation using pattern $C[SA]$.

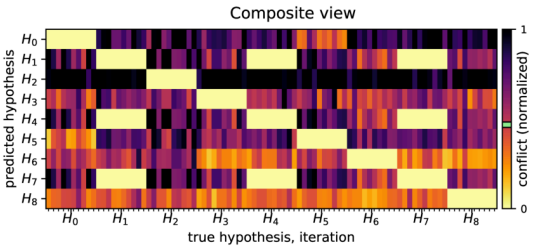


(c) Evaluation using pattern $C[R]$.

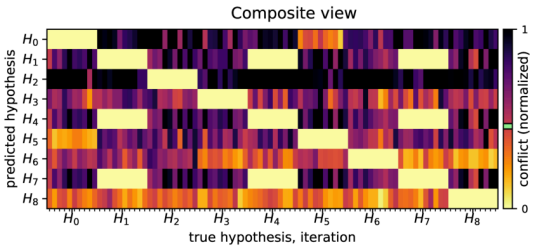


(d) Evaluation using pattern $C[RA]$.

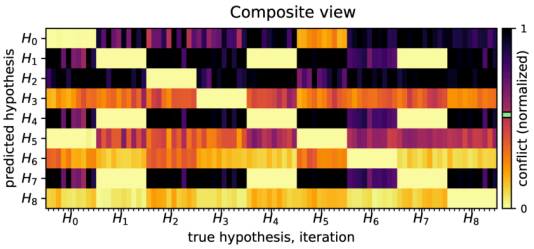
Figure D.15: Pattern and step size evaluation on $H_1^{\ominus}SE(2)$ utilizing Δ_{NW} with a step size of 0.5.



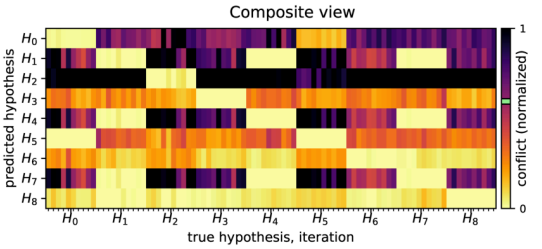
(a) Evaluation using pattern $C[S]$.



(b) Evaluation using pattern $C[SA]$.

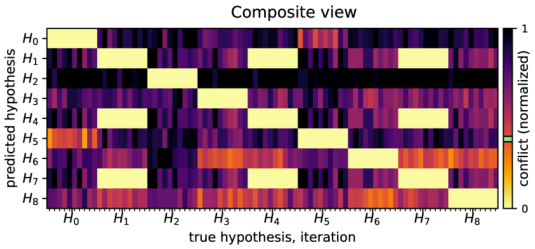


(c) Evaluation using pattern $C[R]$.

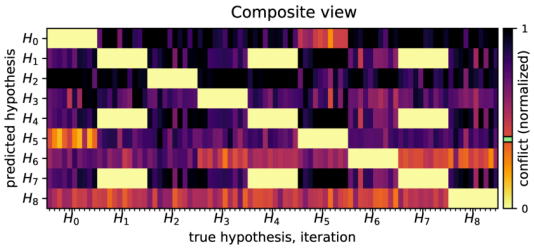


(d) $C[RA]$

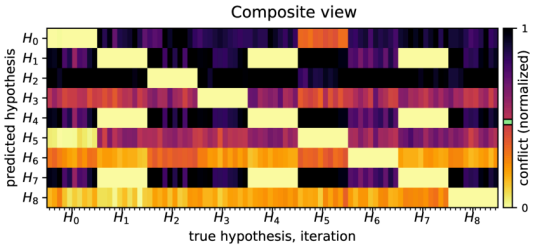
Figure D.16: Pattern and step size evaluation on $H_1^{\ominus}SE(2)$ utilizing Δ_{NW} with a step size of 1.0.



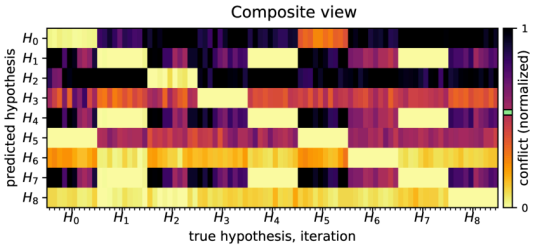
(a) Evaluation using pattern $C[S]$.



(b) Evaluation using pattern $C[SA]$.

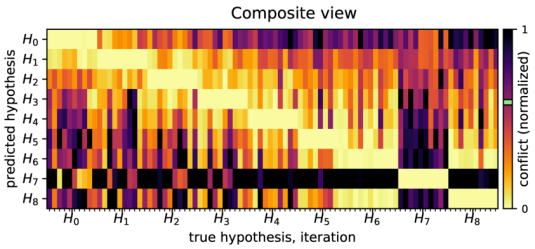


(c) Evaluation using pattern $C[R]$.

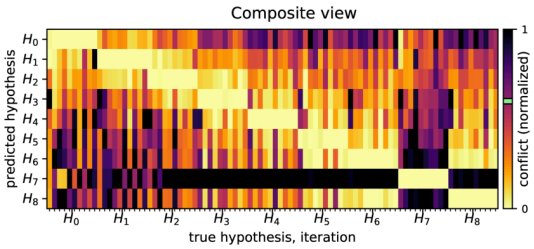


(d) Evaluation using pattern $C[RA]$.

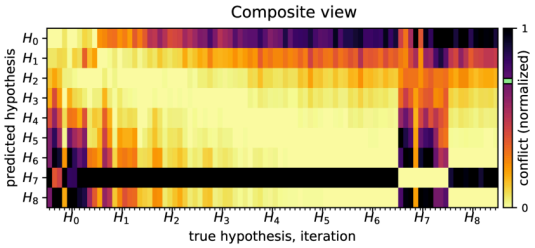
Figure D.17: Pattern and step size evaluation on $\mathrm{H}_1^{\oplus}\mathrm{SE}(2)$ utilizing Δ_{NW} with a step size of 1.5.



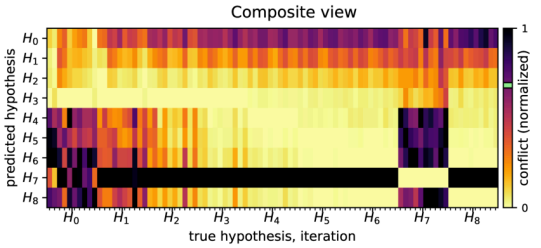
(a) Evaluation using pattern $C[S]$.



(b) Evaluation using pattern $C[SA]$.

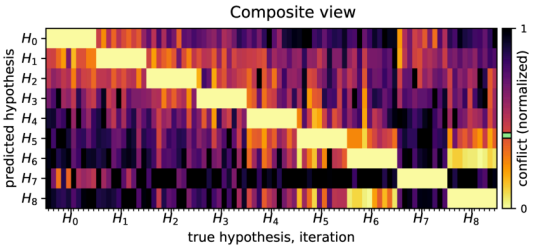


(c) Evaluation using pattern $C[R]$.

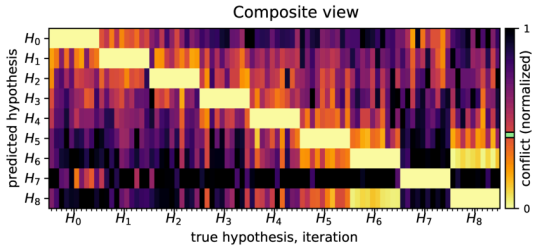


(d) Evaluation using pattern $C[RA]$.

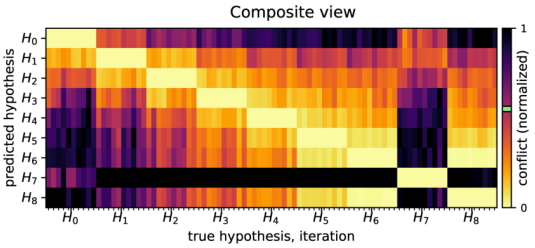
Figure D.18: Pattern and step size evaluation on $\mathrm{H}_2^{\oplus}\mathrm{SE}(2)$ utilizing Δ_{NW} with a step size of 0.5.



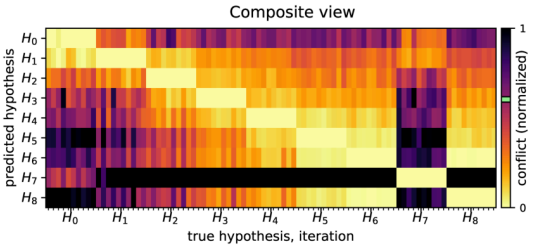
(a) Evaluation using pattern $C[S]$.



(b) Evaluation using pattern $C[SA]$.

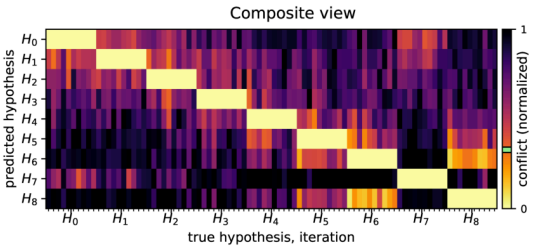


(c) Evaluation using pattern $C[R]$.

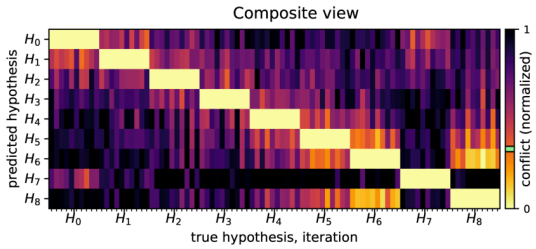


(d) Evaluation using pattern $C[RA]$.

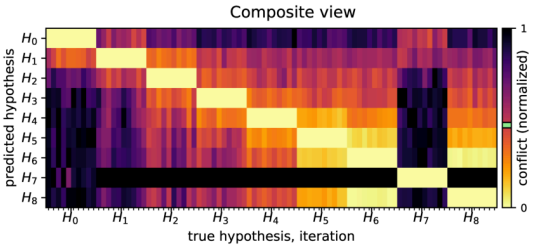
Figure D.19: Pattern and step size evaluation on $\mathrm{H}_2^{\ominus}\mathrm{SE}(2)$ utilizing Δ_{NW} with a step size of 1.0.



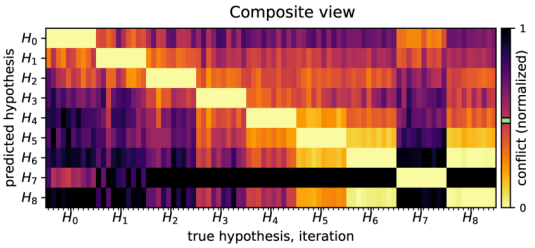
(a) Evaluation using pattern $C[S]$.



(b) Evaluation using pattern $C[SA]$.



(c) Evaluation using pattern $C[R]$.



(d) Evaluation using pattern $C[RA]$.

Figure D.20: Pattern and step size evaluation on $\mathrm{H}_2^{\ominus}\mathrm{SE}(2)$ utilizing Δ_{NW} with a step size of 1.5.

Stepsize	Pattern	$\Delta[\Xi]/\sum I$		Accuracy
-	-	μ	σ	-
0.5	$C[S]$	0.583	0.363	1.00
0.5	$C[SA]$	0.563	0.363	0.96
0.5	$C[R]$	0.625	0.399	0.63
0.5	$C[RA]$	0.616	0.410	0.64
1.0	$C[S]$	0.468	0.340	1.00
1.0	$C[SA]$	0.459	0.348	1.00
1.0	$C[R]$	0.520	0.395	0.96
1.0	$C[RA]$	0.595	0.372	0.74
1.5	$C[S]$	0.386	0.339	1.00
1.5	$C[SA]$	0.384	0.345	1.00
1.5	$C[R]$	0.478	0.377	0.99
1.5	$C[RA]$	0.531	0.381	0.81

(a) Pattern and step size evaluation on $H_1^{\ominus}SE(2)$.

Stepsize	Pattern	$\Delta[\Xi]/\sum I$		Accuracy
-	-	μ	σ	-
0.5	$C[S]$	0.593	0.342	0.92
0.5	$C[SA]$	0.599	0.338	0.88
0.5	$C[R]$	0.706	0.357	0.62
0.5	$C[RA]$	0.682	0.345	0.60
1.0	$C[S]$	0.405	0.313	1.00
1.0	$C[SA]$	0.408	0.318	1.00
1.0	$C[R]$	0.550	0.344	1.00
1.0	$C[RA]$	0.603	0.329	0.94
1.5	$C[S]$	0.328	0.306	1.00
1.5	$C[SA]$	0.332	0.307	1.00
1.5	$C[R]$	0.460	0.326	1.00
1.5	$C[RA]$	0.486	0.311	0.93

(b) Pattern and step size evaluation on $H_2^{\ominus}SE(2)$.

Table D.5: Pattern and step size evaluation on $H_1^{\ominus}SE(2)$ and $H_2^{\ominus}SE(2)$ utilizing Δ_{NW} with various patterns and step sizes. Intrinsic conflict (μ and σ) and classification accuracy.

Stepsize	Pattern	$\Lambda[\Xi]$		$\mathcal{S}$		$\mathcal{B}$		T_{CPU}		T_{WALL}		$\mathcal{S}/\Lambda[\Xi]$		$\mathcal{B}/\Lambda[\Xi]$	
-	-	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
0.5	$C[S]$	125.8	9.7	1861.1	32.7	1923.2	43.3	100.158	8.018	1.859	0.153	14.9	1.4	15.4	1.4
0.5	$C[SA]$	114.8	10.8	1841.2	35.9	1869.0	28.2	90.915	3.848	1.730	0.074	16.2	1.8	16.4	1.8
0.5	$C[R]$	83.4	4.1	4053.6	78.5	4056.8	78.7	297.382	19.675	5.264	0.267	48.8	3.0	48.8	3.0
0.5	$C[RA]$	76.0	3.4	3961.9	74.9	3965.4	75.9	310.110	17.735	5.425	0.248	52.2	3.0	52.3	3.0
1.0	$C[S]$	106.4	8.2	1960.2	27.4	2124.6	37.2	128.800	7.787	2.536	0.155	18.5	1.6	20.1	1.8
1.0	$C[SA]$	99.9	5.9	1901.5	21.7	2002.0	17.8	117.709	4.152	2.349	0.077	19.1	1.3	20.1	1.2
1.0	$C[R]$	39.1	1.5	2110.7	20.2	2138.7	23.1	214.282	16.917	3.656	0.244	54.1	2.4	54.8	2.5
1.0	$C[RA]$	70.3	2.8	4089.1	60.8	4122.7	64.5	446.723	33.705	7.417	0.476	58.2	2.7	58.7	2.7
1.5	$C[S]$	92.7	4.5	2051.6	17.8	2314.9	30.1	166.040	8.028	3.303	0.128	22.2	1.2	25.0	1.2
1.5	$C[SA]$	88.3	5.2	1981.3	26.4	2155.3	33.5	148.136	4.780	3.002	0.084	22.5	1.5	24.5	1.5
1.5	$C[R]$	34.5	1.1	2196.8	23.0	2277.6	24.8	322.562	21.039	5.212	0.277	63.7	2.4	66.1	2.6
1.5	$C[RA]$	60.9	2.2	4307.5	38.0	4402.2	40.6	696.122	34.371	10.880	0.440	70.8	2.6	72.3	2.6

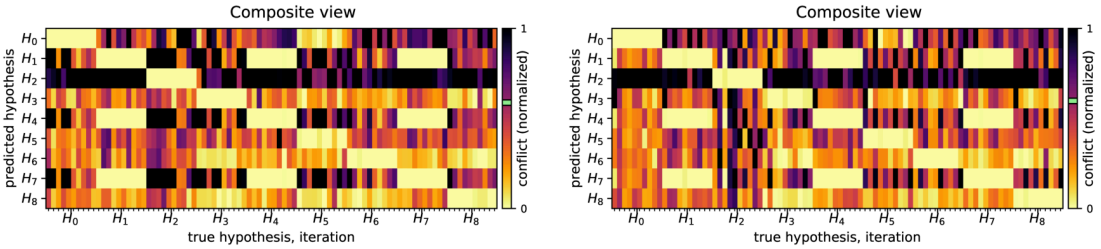
(a) Pattern and step size evaluation on $H_1^{\ominus}SE(2)$.

Stepsize	Pattern	$\Lambda[\Xi]$		$\mathcal{S}$		$\mathcal{B}$		T_{CPU}		T_{WALL}		$\mathcal{S}/\Lambda[\Xi]$		$\mathcal{B}/\Lambda[\Xi]$	
-	-	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
0.5	$C[S]$	134.9	8.6	1835.9	27.0	1888.6	40.2	97.984	6.482	1.790	0.143	13.7	1.0	14.1	1.1
0.5	$C[SA]$	122.8	10.3	1812.3	33.9	1845.4	26.9	90.867	3.993	1.710	0.051	14.9	1.5	15.1	1.4
0.5	$C[R]$	93.0	4.0	4039.2	111.5	4039.2	111.5	287.248	18.364	5.126	0.254	43.5	2.9	43.5	2.9
0.5	$C[RA]$	82.4	3.2	3839.4	76.7	3839.4	76.7	291.511	19.835	5.152	0.259	46.7	2.5	46.7	2.5
1.0	$C[S]$	112.2	8.8	1938.5	29.5	2086.5	60.4	116.480	10.352	2.343	0.218	17.4	1.6	18.7	1.8
1.0	$C[SA]$	107.1	5.8	1891.9	23.2	2014.5	31.6	115.025	7.287	2.319	0.145	17.7	1.2	18.9	1.2
1.0	$C[R]$	19.9	0.6	1045.2	16.6	1051.7	16.5	82.652	4.855	1.452	0.060	52.5	2.4	52.8	2.4
1.0	$C[RA]$	25.7	1.6	1343.0	16.5	1350.6	19.7	109.339	9.910	1.910	0.140	52.5	3.7	52.8	3.8
1.5	$C[S]$	92.2	4.4	2021.4	25.1	2279.4	36.8	146.102	9.665	3.039	0.158	22.0	1.3	24.8	1.3
1.5	$C[SA]$	88.2	6.3	1977.0	32.1	2201.1	42.9	145.477	10.503	3.009	0.174	22.6	2.0	25.1	2.1
1.5	$C[R]$	9.9	0.3	611.3	6.9	625.7	6.3	70.405	5.187	1.196	0.070	61.9	2.4	63.3	2.4
1.5	$C[RA]$	12.3	0.6	854.6	9.6	870.3	11.4	105.543	9.937	1.754	0.144	69.6	3.6	70.9	3.9

(b) Pattern and step size evaluation on $H_2^{\ominus}SE(2)$.

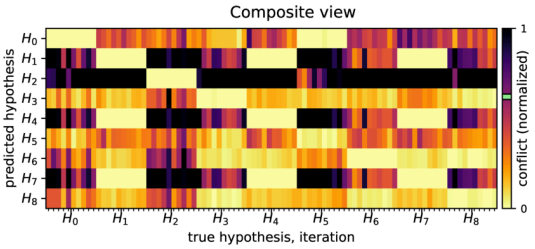
Table D.6: Pattern and step size evaluation on $H_1^{\ominus}SE(2)$ and $H_2^{\ominus}SE(2)$ utilizing Δ_{NW} with various patterns and step sizes. Sensorimotor map properties and timing data.

D.3.2 Pattern and Step size Evaluation on $H_1^{\ominus}SE(2)$ and $H_2^{\ominus}SE(2)$ utilizing Δ_{RBFN}

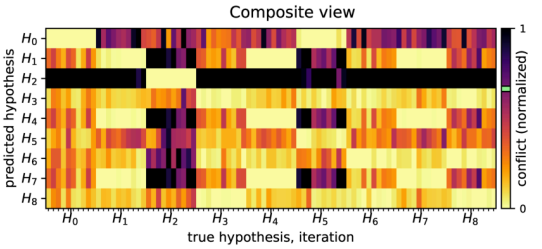


(a) Evaluation using pattern $C[S]$.

(b) Evaluation using pattern $C[SA]$.

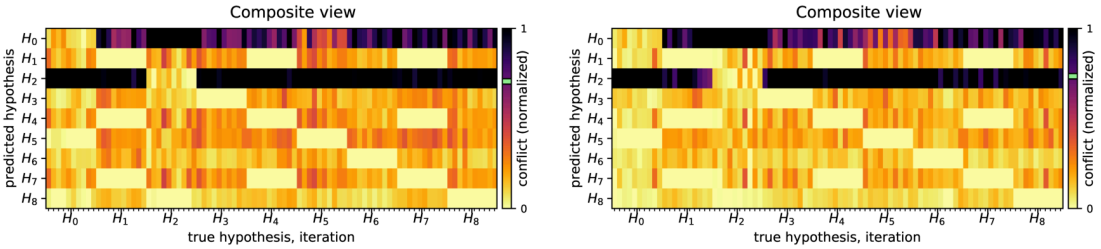


(c) Evaluation using pattern $C[R]$.



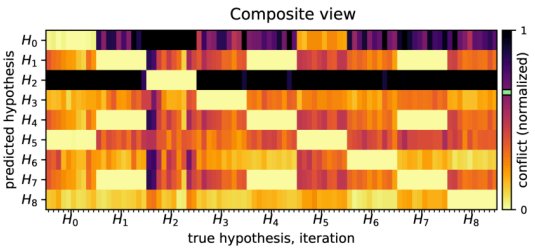
(d) Evaluation using pattern $C[RA]$.

Figure D.21: Pattern and step size evaluation on $H_1^{\ominus}SE(2)$ utilizing Δ_{RBFN} with a step size of 0.5.

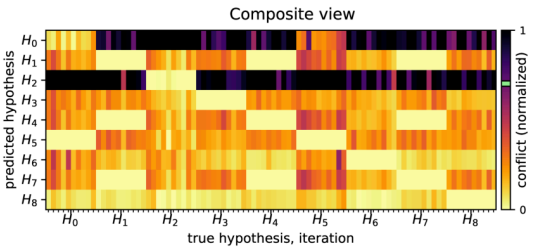


(a) Evaluation using pattern $C[S]$.

(b) Evaluation using pattern $C[SA]$.

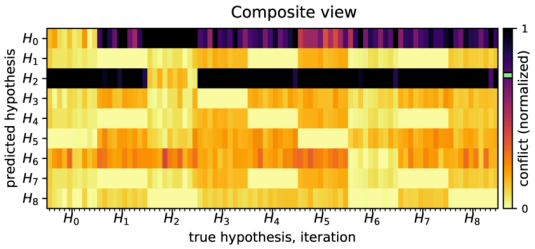


(c) Evaluation using pattern $C[R]$.

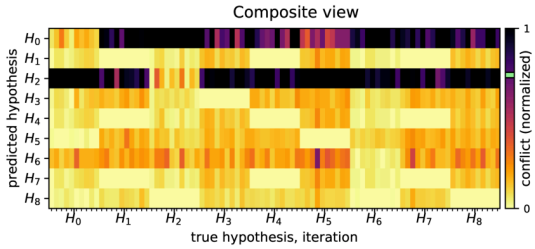


(d) Evaluation using pattern $C[RA]$.

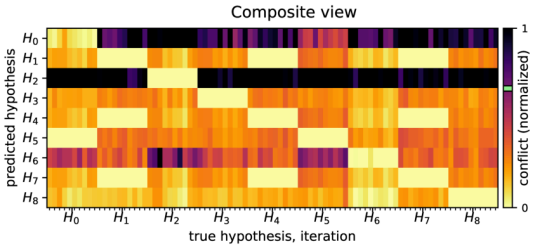
Figure D.22: Pattern and step size evaluation on $H_1^{\ominus}SE(2)$ utilizing Δ_{RBFN} with a step size of 1.0.



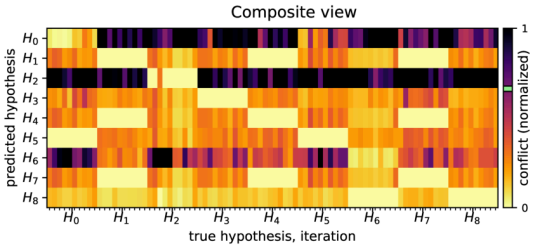
(a) Evaluation using pattern $C[S]$.



(b) Evaluation using pattern $C[SA]$.

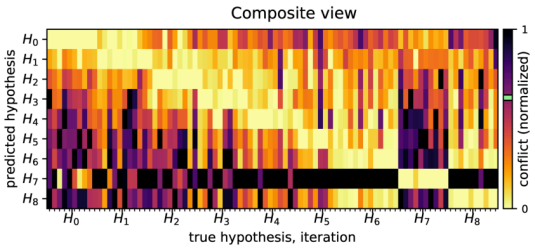


(c) Evaluation using pattern $C[R]$.

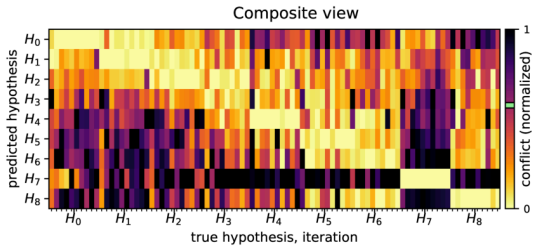


(d) Evaluation using pattern $C[RA]$.

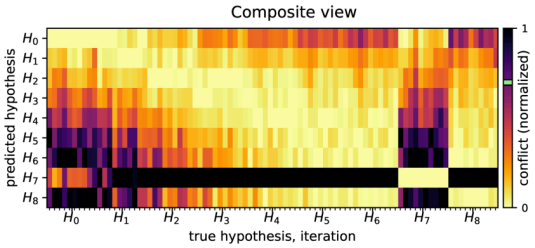
Figure D.23: Pattern and step size evaluation on $\mathrm{H}_1^{\ominus}\mathrm{SE}(2)$ utilizing Δ_{RBFN} with a step size of 1.5.



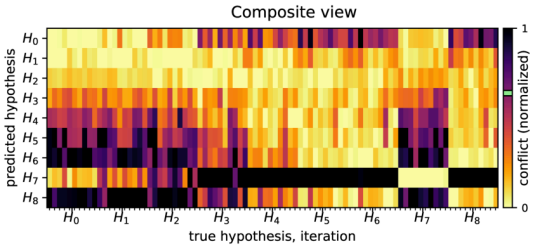
(a) Evaluation using pattern $C[S]$.



(b) Evaluation using pattern $C[SA]$.

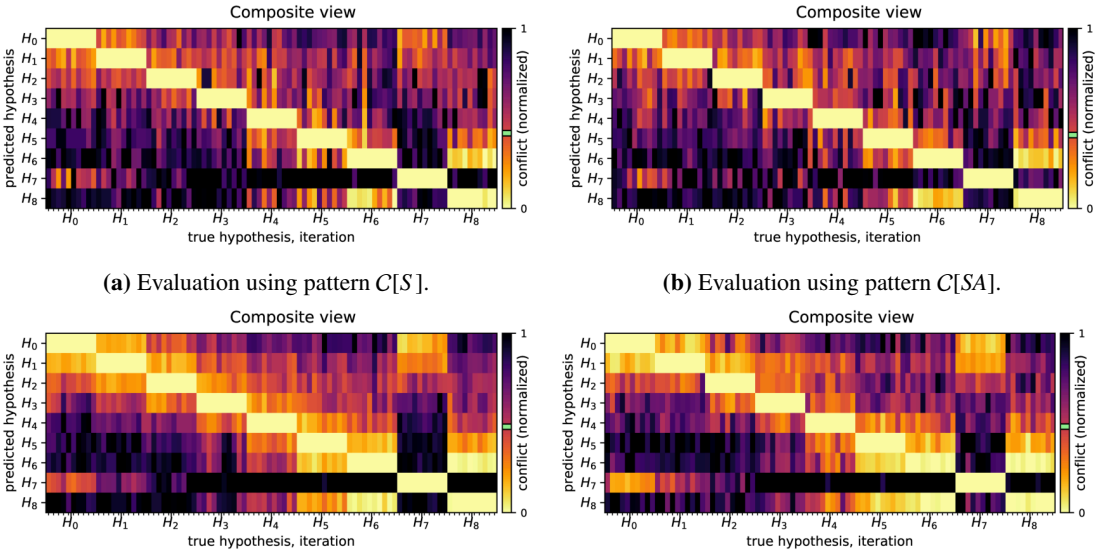


(c) Evaluation using pattern $C[R]$.



(d) Evaluation using pattern $C[RA]$.

Figure D.24: Pattern and step size evaluation on $\mathrm{H}_2^{\ominus}\mathrm{SE}(2)$ utilizing Δ_{RBFN} with a step size of 0.5.



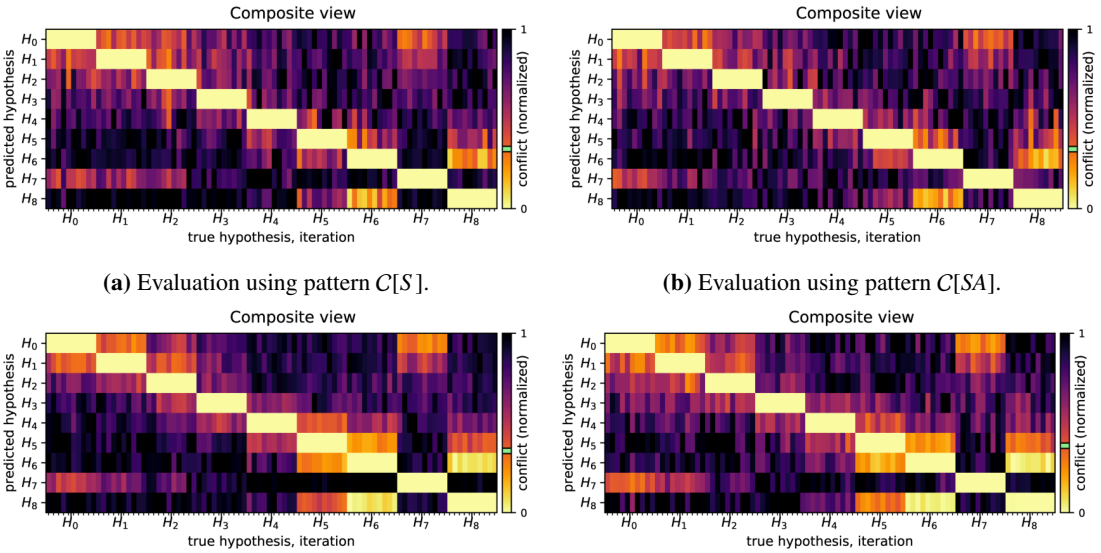
(a) Evaluation using pattern $C[S]$.

(b) Evaluation using pattern $C[SA]$.

(c) Evaluation using pattern $C[R]$.

(d) Evaluation using pattern $C[RA]$.

Figure D.25: Pattern and step size evaluation on $\mathrm{H}_2^{\ominus}\mathrm{SE}(2)$ utilizing Δ_{RBFN} with a step size of 1.0.



(a) Evaluation using pattern $C[S]$.

(b) Evaluation using pattern $C[SA]$.

(c) Evaluation using pattern $C[R]$.

(d) Evaluation using pattern $C[RA]$.

Figure D.26: Pattern and step size evaluation on $\mathrm{H}_2^{\ominus}\mathrm{SE}(2)$ utilizing Δ_{RBFN} with a step size of 1.5.

Stepsize	Pattern	$\Delta[\Xi]/\sum I$		Accuracy
-	-	μ	σ	-
0.5	$C[S]$	0.588	0.358	0.89
0.5	$C[SA]$	0.598	0.343	0.88
0.5	$C[R]$	0.620	0.362	0.83
0.5	$C[RA]$	0.664	0.345	0.76
1.0	$C[S]$	0.706	0.324	0.81
1.0	$C[SA]$	0.734	0.331	0.79
1.0	$C[R]$	0.654	0.315	0.91
1.0	$C[RA]$	0.703	0.327	0.84
1.5	$C[S]$	0.739	0.332	0.70
1.5	$C[SA]$	0.741	0.337	0.68
1.5	$C[R]$	0.665	0.323	0.86
1.5	$C[RA]$	0.663	0.324	0.77

(a) Pattern and step size evaluation on $H_1^{\ominus}SE(2)$.

Stepsize	Pattern	$\Delta[\Xi]/\sum I$		Accuracy
-	-	μ	σ	-
0.5	$C[S]$	0.619	0.334	0.71
0.5	$C[SA]$	0.576	0.344	0.71
0.5	$C[R]$	0.698	0.338	0.53
0.5	$C[RA]$	0.638	0.344	0.44
1.0	$C[S]$	0.418	0.316	0.99
1.0	$C[SA]$	0.408	0.316	0.98
1.0	$C[R]$	0.479	0.332	0.98
1.0	$C[RA]$	0.480	0.332	0.90
1.5	$C[S]$	0.333	0.307	1.00
1.5	$C[SA]$	0.332	0.304	1.00
1.5	$C[R]$	0.344	0.326	1.00
1.5	$C[RA]$	0.373	0.324	0.99

(b) Pattern and step size evaluation on $H_2^{\ominus}SE(2)$.

Table D.7: Pattern and step size evaluation on $H_1^{\ominus}SE(2)$ and $H_2^{\ominus}SE(2)$ utilizing Δ_{RBFN} with various patterns and step sizes. Intrinsic conflict (μ and σ) and classification accuracy.

Stepsize	Pattern	$\Lambda[\Xi]$		$\mathcal{S}$		$\mathcal{B}$		T_{CPU}		T_{WALL}		$\mathcal{S}/\Lambda[\Xi]$		$\mathcal{B}/\Lambda[\Xi]$	
-	-	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
0.5	$C[S]$	125.8	9.7	1861.1	32.7	1923.2	43.3	104.311	5.835	1.720	0.120	14.9	1.4	15.4	1.4
0.5	$C[SA]$	114.8	10.8	1841.2	35.9	1869.0	28.2	94.201	5.051	1.570	0.065	16.2	1.8	16.4	1.8
0.5	$C[R]$	83.4	4.1	4053.6	78.5	4056.8	78.7	219.328	10.706	5.068	0.307	48.8	3.0	48.8	3.0
0.5	$C[RA]$	76.0	3.4	3961.9	74.9	3965.4	75.9	225.612	9.216	5.212	0.238	52.2	3.0	52.3	3.0
1.0	$C[S]$	106.4	8.2	1960.2	27.4	2124.6	37.2	117.994	4.573	2.201	0.142	18.5	1.6	20.1	1.8
1.0	$C[SA]$	99.9	5.9	1901.5	21.7	2002.0	17.8	108.279	4.387	2.019	0.065	19.1	1.3	20.1	1.2
1.0	$C[R]$	39.1	1.5	2110.7	20.2	2138.7	23.1	147.896	9.714	3.521	0.254	54.1	2.4	54.8	2.5
1.0	$C[RA]$	70.3	2.8	4089.1	60.8	4122.7	64.5	308.047	19.574	7.249	0.491	58.2	2.7	58.7	2.7
1.5	$C[S]$	92.7	4.5	2051.6	17.8	2314.9	30.1	137.066	5.572	2.879	0.149	22.2	1.2	25.0	1.2
1.5	$C[SA]$	88.3	5.2	1981.3	26.4	2155.3	33.5	124.692	3.536	2.574	0.079	22.5	1.5	24.5	1.5
1.5	$C[R]$	34.5	1.1	2196.8	23.0	2277.6	24.8	213.825	13.106	5.121	0.302	63.7	2.4	66.1	2.6
1.5	$C[RA]$	60.9	2.2	4307.5	38.0	4402.2	40.6	457.338	23.590	10.610	0.465	70.8	2.6	72.3	2.6

(a) Pattern and step size evaluation on $H_1^{\ominus}SE(2)$.

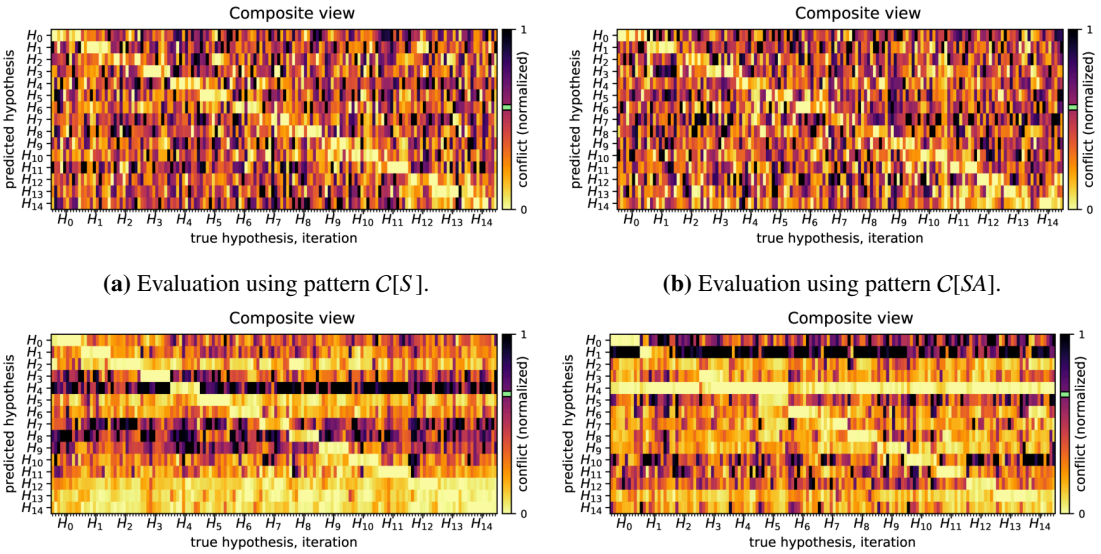
Stepsize	Pattern	$\Lambda[\Xi]$		$\mathcal{S}$		$\mathcal{B}$		T_{CPU}		T_{WALL}		$\mathcal{S}/\Lambda[\Xi]$		$\mathcal{B}/\Lambda[\Xi]$	
-	-	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
0.5	$C[S]$	134.9	8.6	1835.9	27.0	1888.6	40.2	106.492	5.170	1.691	0.110	13.7	1.0	14.1	1.1
0.5	$C[SA]$	122.8	10.3	1812.3	33.9	1845.4	26.9	98.421	4.881	1.608	0.037	14.9	1.5	15.1	1.4
0.5	$C[R]$	93.0	4.0	4039.2	111.5	4039.2	111.5	217.201	9.688	4.860	0.292	43.5	2.9	43.5	2.9
0.5	$C[RA]$	82.4	3.2	3839.4	76.7	3839.4	76.7	217.136	11.706	4.892	0.292	46.7	2.5	46.7	2.5
1.0	$C[S]$	112.2	8.8	1938.5	29.5	2086.5	60.4	115.101	6.323	2.054	0.153	17.4	1.6	18.7	1.8
1.0	$C[SA]$	107.1	5.8	1891.9	23.2	2014.5	31.6	111.828	5.412	2.029	0.113	17.7	1.2	18.9	1.2
1.0	$C[R]$	19.9	0.6	1045.2	16.6	1051.7	16.5	58.788	2.651	1.366	0.069	52.5	2.4	52.8	2.4
1.0	$C[RA]$	25.7	1.6	1343.0	16.5	1350.6	19.7	78.385	5.072	1.807	0.143	52.5	3.7	52.8	3.8
1.5	$C[S]$	92.2	4.4	2021.4	25.1	2279.4	36.8	127.090	5.130	2.512	0.139	22.0	1.3	24.8	1.3
1.5	$C[SA]$	88.2	6.3	1977.0	32.1	2201.1	42.9	124.516	4.320	2.506	0.152	22.6	2.0	25.1	2.1
1.5	$C[R]$	9.9	0.3	611.3	6.9	625.7	6.3	46.871	2.930	1.145	0.078	61.9	2.4	63.3	2.4
1.5	$C[RA]$	12.3	0.6	854.6	9.6	870.3	11.4	69.560	5.911	1.698	0.151	69.6	3.6	70.9	3.9

(b) Pattern and step size evaluation on $H_2^{\ominus}SE(2)$.

Table D.8: Pattern and step size evaluation on $H_1^{\ominus}SE(2)$ and $H_2^{\ominus}SE(2)$ utilizing Δ_{RBFN} with various patterns and step sizes. Sensorimotor map properties and timing data.

D.4 Scenario IIIa : Holonomic Pattern Evaluation on SE(3)

D.4.1 Pattern and Step size Evaluation on H$^{\oplus}$SE(3) utilizing Δ_{NW}



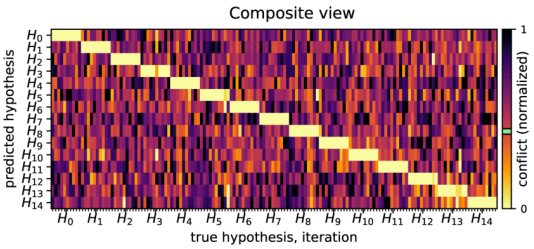
(a) Evaluation using pattern $C[S]$.

(b) Evaluation using pattern $C[SA]$.

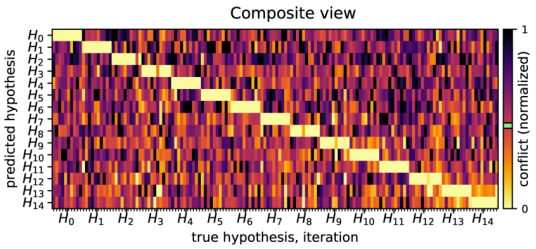
(c) Evaluation using pattern $C[R]$.

(d) Evaluation using pattern $C[RA]$.

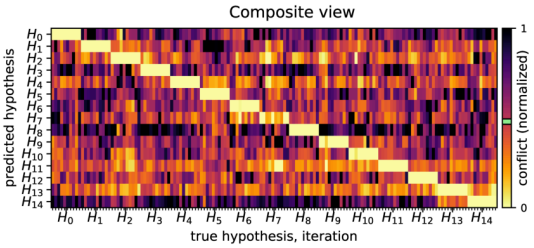
Figure D.27: Pattern and step size evaluation on H$^{\oplus}$SE(3) utilizing Δ_{NW} with a step size of 0.5.



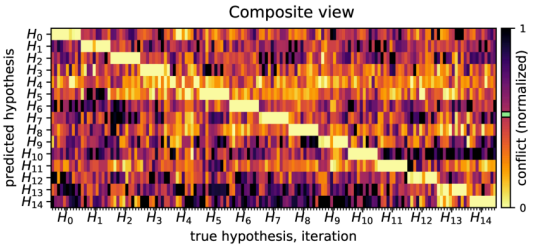
(a) Evaluation using pattern $C[S]$.



(b) Evaluation using pattern $C[SA]$.

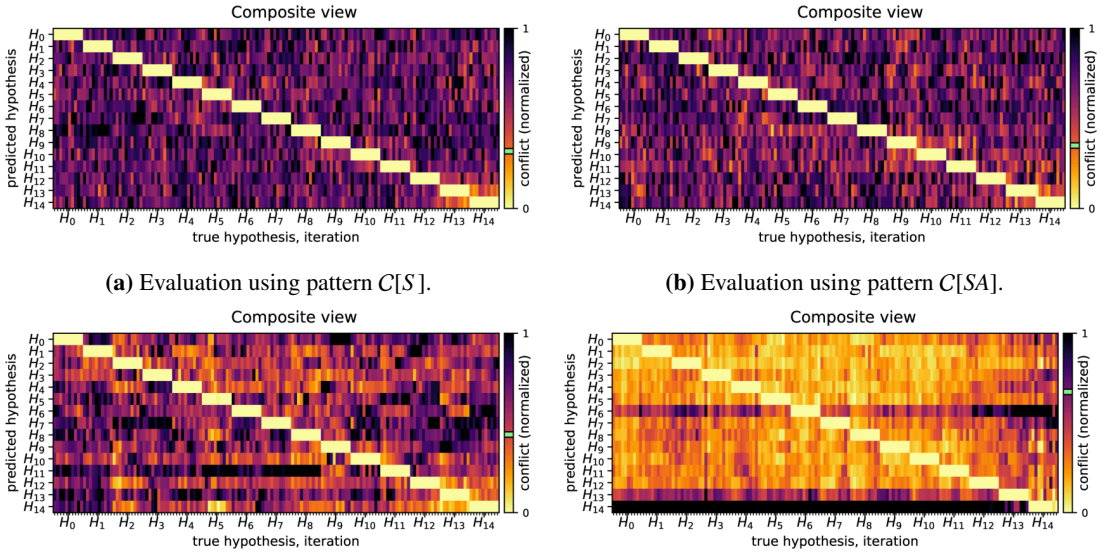


(c) Evaluation using pattern $C[R]$.



(d) Evaluation using pattern $C[RA]$.

Figure D.28: Pattern and step size evaluation on H$^{\oplus}$SE(3) utilizing Δ_{NW} with a step size of 1.0.



(a) Evaluation using pattern $C[S]$.

(b) Evaluation using pattern $C[SA]$.

(c) Evaluation using pattern $C[R]$.

(d) Evaluation using pattern $C[RA]$.

Figure D.29: Pattern and step size evaluation on $H^{\oplus}SE(3)$ utilizing Δ_{NW} with a step size of 1.5.

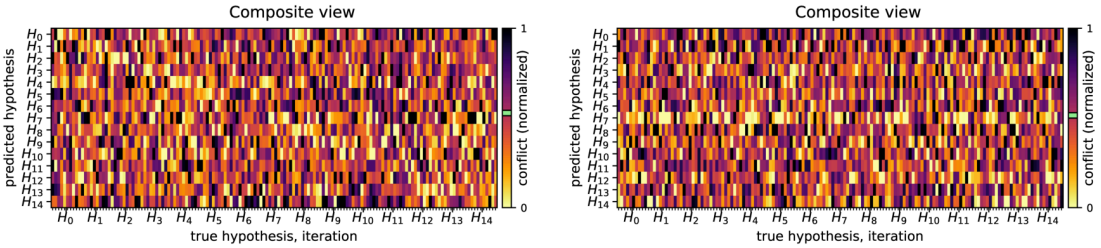
Stepsize	Pattern	$\Delta[\Xi]/\sum I$		Accuracy
-	-	μ	σ	-
0.5	$C[S]$	0.568	0.287	0.56
0.5	$C[SA]$	0.567	0.287	0.44
0.5	$C[R]$	0.667	0.294	0.65
0.5	$C[RA]$	0.664	0.291	0.43
1.0	$C[S]$	0.431	0.258	0.95
1.0	$C[SA]$	0.463	0.268	0.93
1.0	$C[R]$	0.479	0.271	0.95
1.0	$C[RA]$	0.519	0.276	0.86
1.5	$C[S]$	0.318	0.241	1.00
1.5	$C[SA]$	0.349	0.245	1.00
1.5	$C[R]$	0.435	0.262	0.99
1.5	$C[RA]$	0.673	0.253	0.97

Table D.9: Pattern and step size evaluation on $H^{\oplus}SE(3)$ utilizing Δ_{NW} with various patterns and step sizes. Intrinsic conflict (μ and σ) and classification accuracy.

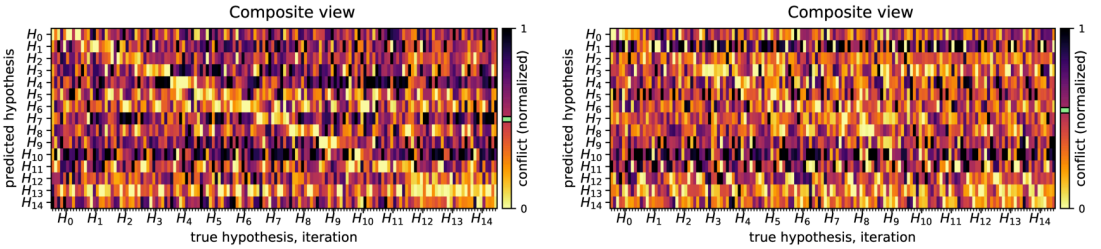
Stepsize	Pattern	$\Lambda[\Xi]$		$\mathcal{S}$		$\mathcal{B}$		T_{CPU}		T_{WALL}		$\mathcal{S}/\Lambda[\Xi]$		$\mathcal{B}/\Lambda[\Xi]$	
-	-	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
0.5	$C[S]$	293.3	4.7	1618.0	15.1	1640.7	16.9	201.267	3.095	2.851	0.041	5.5	0.1	5.6	0.1
0.5	$C[SA]$	276.6	6.1	1735.7	8.6	1756.6	9.8	193.892	3.901	2.761	0.089	6.3	0.2	6.4	0.2
0.5	$C[R]$	283.4	1.8	3531.6	13.5	3531.6	13.5	388.758	2.675	6.242	0.049	12.5	0.1	12.5	0.1
0.5	$C[RA]$	260.0	1.5	3638.5	26.3	3638.5	26.3	357.673	2.181	5.507	0.031	14.0	0.1	14.0	0.1
1.0	$C[S]$	272.5	4.6	1662.0	14.7	1768.1	18.0	198.913	3.921	2.906	0.068	6.1	0.1	6.5	0.1
1.0	$C[SA]$	264.2	5.6	1730.6	11.3	1819.5	17.1	193.532	3.242	2.790	0.051	6.6	0.2	6.9	0.2
1.0	$C[R]$	262.0	2.4	3686.5	17.3	3689.9	17.4	385.900	3.714	6.183	0.120	14.1	0.1	14.1	0.1
1.0	$C[RA]$	247.4	1.3	3714.7	17.3	3719.3	18.0	353.633	2.914	5.786	0.040	15.0	0.1	15.0	0.1
1.5	$C[S]$	238.0	4.1	1773.7	13.7	1979.7	17.5	198.293	1.398	3.157	0.049	7.5	0.2	8.3	0.2
1.5	$C[SA]$	233.8	5.2	1816.4	13.0	1975.8	14.9	192.626	3.147	3.069	0.036	7.8	0.2	8.5	0.2
1.5	$C[R]$	105.2	0.9	2052.0	8.1	2065.1	7.9	194.999	1.689	3.596	0.035	19.5	0.2	19.6	0.2
1.5	$C[RA]$	201.9	1.3	4104.5	22.3	4121.1	24.4	364.884	3.306	6.892	0.075	20.3	0.2	20.4	0.2

Table D.10: Pattern and step size evaluation on $H^{\oplus}SE(3)$ utilizing Δ_{NW} with various patterns and step sizes. Sensorimotor map properties and timing data.

D.4.2 Pattern and Step size Evaluation on $H^{\oplus}SE(3)$ utilizing Δ_{RBFN}

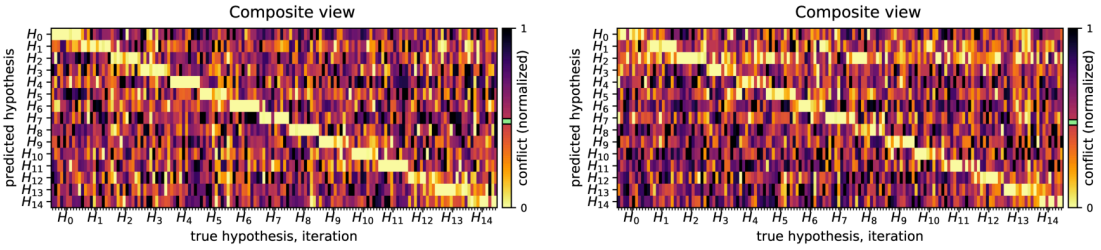


(a) Evaluation using pattern $C[S]$. **(b)** Evaluation using pattern $C[SA]$.

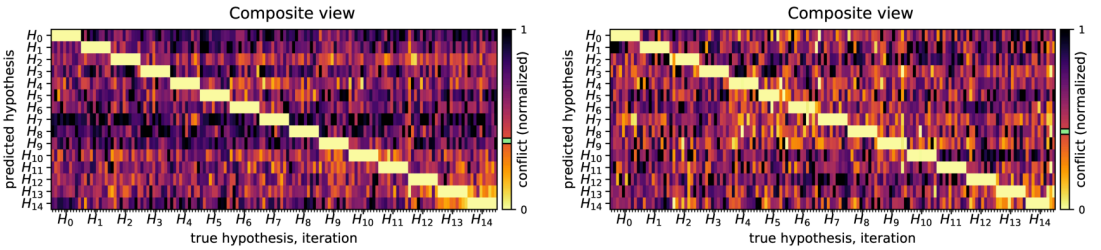


(c) Evaluation using pattern $C[R]$. **(d)** Evaluation using pattern $C[RA]$.

Figure D.30: Pattern and step size evaluation on $H^{\oplus}SE(3)$ utilizing Δ_{RBFN} with a step size of 0.5.

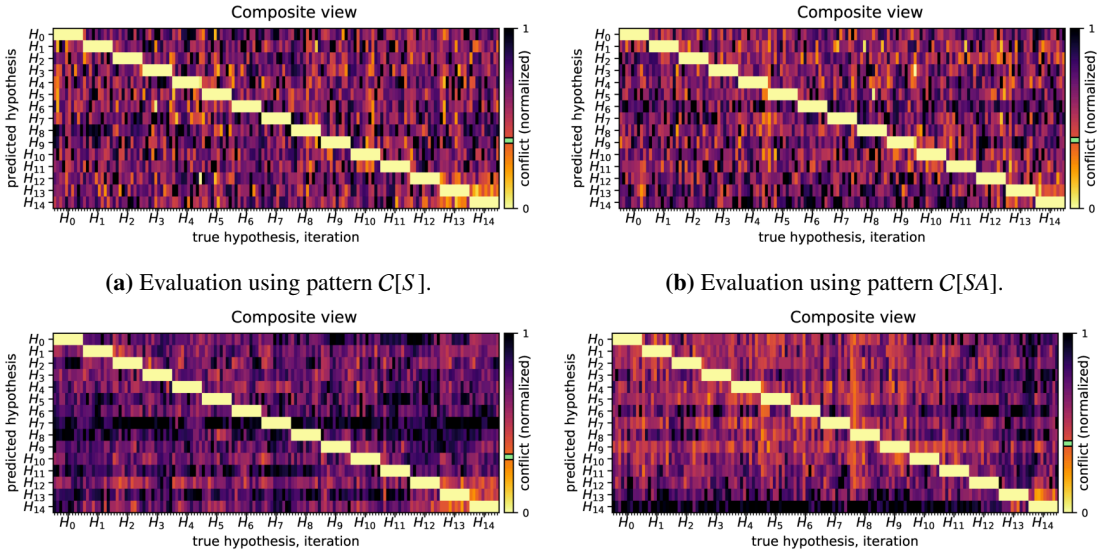


(a) Evaluation using pattern $C[S]$. **(b)** Evaluation using pattern $C[SA]$.



(c) Evaluation using pattern $C[R]$. **(d)** Evaluation using pattern $C[RA]$.

Figure D.31: Pattern and step size evaluation on $H^{\oplus}SE(3)$ utilizing Δ_{RBFN} with a step size of 1.0.



(a) Evaluation using pattern $C[S]$.

(b) Evaluation using pattern $C[SA]$.

(c) Evaluation using pattern $C[R]$.

(d) Evaluation using pattern $C[RA]$.

Figure D.32: Pattern and step size evaluation on $H^{\oplus}SE(3)$ utilizing Δ_{RBFN} with a step size of 1.5.

Stepsize	Pattern	$\Delta[\Xi]/\sum I$		Accuracy
-	-	μ	σ	-
0.5	$C[S]$	0.527	0.287	0.21
0.5	$C[SA]$	0.514	0.289	0.12
0.5	$C[R]$	0.496	0.291	0.33
0.5	$C[RA]$	0.545	0.287	0.23
1.0	$C[S]$	0.481	0.285	0.72
1.0	$C[SA]$	0.475	0.284	0.59
1.0	$C[R]$	0.381	0.255	1.00
1.0	$C[RA]$	0.433	0.265	0.93
1.5	$C[S]$	0.378	0.253	0.99
1.5	$C[SA]$	0.379	0.249	0.99
1.5	$C[R]$	0.311	0.239	1.00
1.5	$C[RA]$	0.386	0.238	1.00

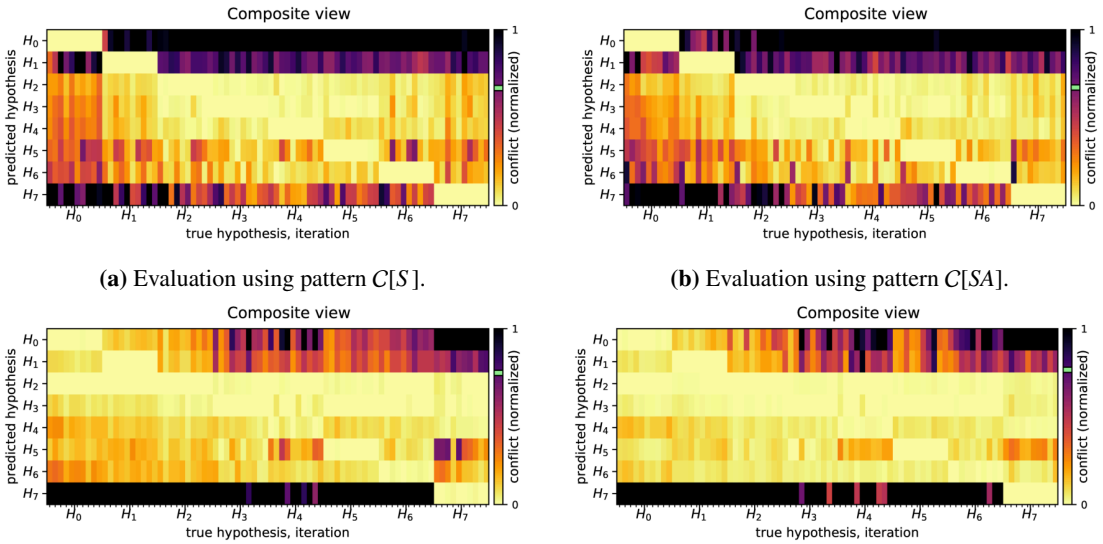
Table D.11: Pattern and step size evaluation on $H^{\oplus}SE(3)$ utilizing Δ_{RBFN} with various patterns and step sizes. Intrinsic conflict (μ and σ) and classification accuracy.

Stepsize	Pattern	$\Lambda[\Xi]$		$\mathcal{S}$		$\mathcal{B}$		T_{CPU}		T_{WALL}		$\mathcal{S}/\Lambda[\Xi]$		$\mathcal{B}/\Lambda[\Xi]$	
-	-	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
0.5	$C[S]$	293.3	4.7	1618.0	15.1	1640.7	16.9	228.372	2.859	3.093	0.034	5.5	0.1	5.6	0.1
0.5	$C[SA]$	276.6	6.1	1735.7	8.6	1756.6	9.8	227.179	6.087	3.058	0.115	6.3	0.2	6.4	0.2
0.5	$C[R]$	283.4	1.8	3531.6	13.5	3531.6	13.5	400.545	1.974	5.960	0.040	12.5	0.1	12.5	0.1
0.5	$C[RA]$	260.0	1.5	3638.5	26.3	3638.5	26.3	374.478	2.550	5.309	0.038	14.0	0.1	14.0	0.1
1.0	$C[S]$	272.5	4.6	1662.0	14.7	1768.1	18.0	224.638	4.487	3.057	0.074	6.1	0.1	6.5	0.1
1.0	$C[SA]$	264.2	5.6	1730.6	11.3	1819.5	17.1	222.827	3.563	2.979	0.059	6.6	0.2	6.9	0.2
1.0	$C[R]$	262.0	2.4	3686.5	17.3	3689.9	17.4	390.869	4.191	5.763	0.125	14.1	0.1	14.1	0.1
1.0	$C[RA]$	247.4	1.3	3714.7	17.3	3719.3	18.0	363.412	3.550	5.453	0.049	15.0	0.1	15.0	0.1
1.5	$C[S]$	238.0	4.1	1773.7	13.7	1979.7	17.5	217.411	1.748	3.076	0.036	7.5	0.2	8.3	0.2
1.5	$C[SA]$	233.8	5.2	1816.4	13.0	1975.8	14.9	217.180	3.142	3.057	0.029	7.8	0.2	8.5	0.2
1.5	$C[R]$	105.2	0.9	2052.0	8.1	2065.1	7.9	177.912	1.248	3.091	0.020	19.5	0.2	19.6	0.2
1.5	$C[RA]$	201.9	1.3	4104.5	22.3	4121.1	24.4	339.679	1.852	5.954	0.053	20.3	0.2	20.4	0.2

Table D.12: Pattern and step size evaluation on $H^{\oplus}SE(3)$ utilizing Δ_{RBFN} with various patterns and step sizes. Sensorimotor map properties and timing data.

D.5 Scenario IIIb : Nonholonomic Pattern Evaluation on SE(3)

D.5.1 Pattern and Step size Evaluation on $H^{\ominus}SE(3)$ utilizing Δ_{NW}



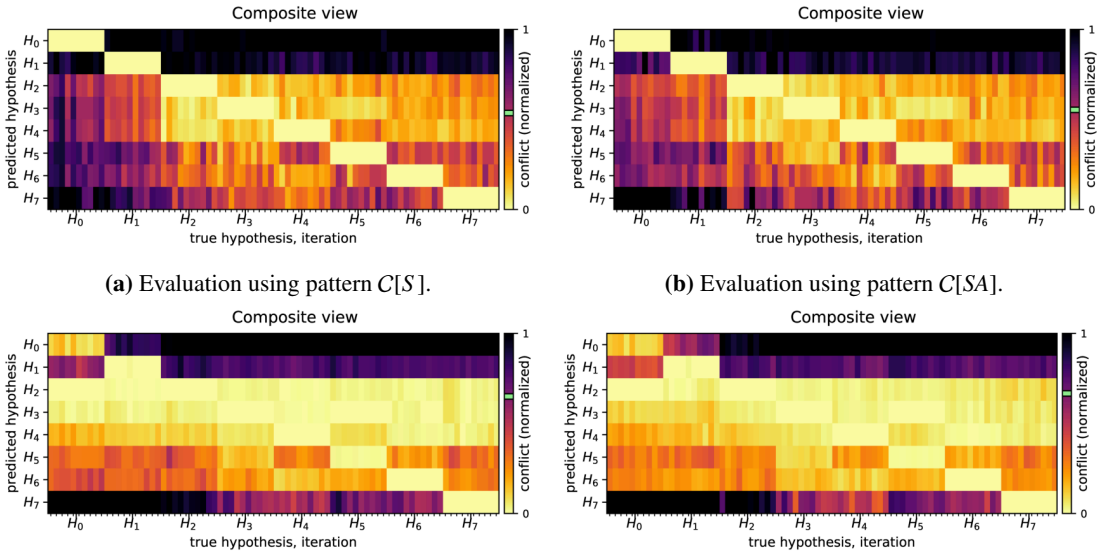
(a) Evaluation using pattern $C[S]$.

(b) Evaluation using pattern $C[SA]$.

(c) Evaluation using pattern $C[R]$.

(d) Evaluation using pattern $C[RA]$.

Figure D.33: Pattern and step size evaluation on $H^{\ominus}SE(3)$ utilizing Δ_{NW} with a step size of 0.5.



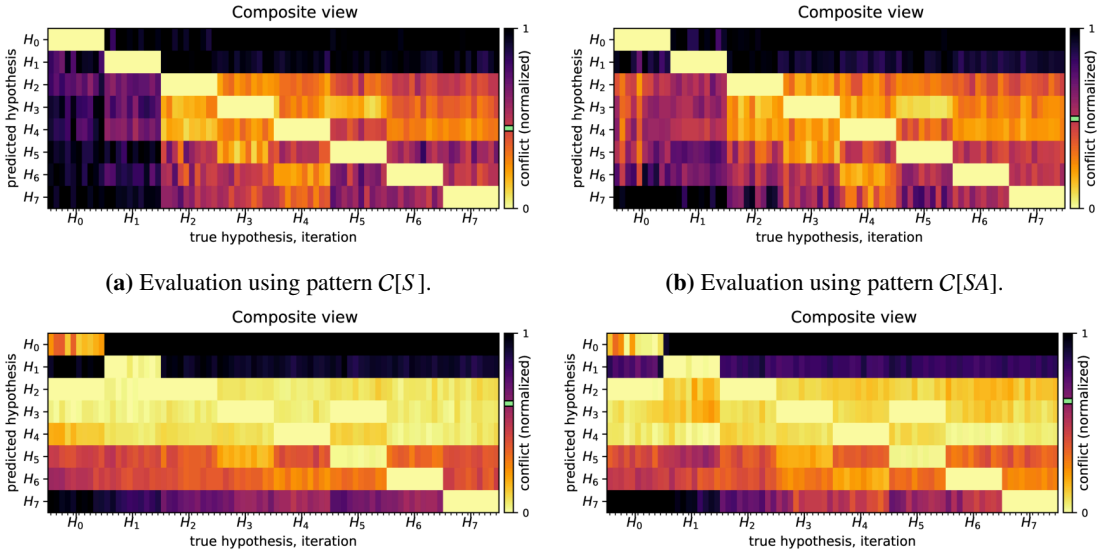
(a) Evaluation using pattern $C[S]$.

(b) Evaluation using pattern $C[SA]$.

(c) Evaluation using pattern $C[R]$.

(d) Evaluation using pattern $C[RA]$.

Figure D.34: Pattern and step size evaluation on $H^{\ominus}SE(3)$ utilizing Δ_{NW} with a step size of 1.0.



(a) Evaluation using pattern $C[S]$.

(b) Evaluation using pattern $C[SA]$.

(c) Evaluation using pattern $C[R]$.

(d) Evaluation using pattern $C[RA]$.

Figure D.35: Pattern and step size evaluation on $H^{\ominus}SE(3)$ utilizing Δ_{NW} with a step size of 1.5.

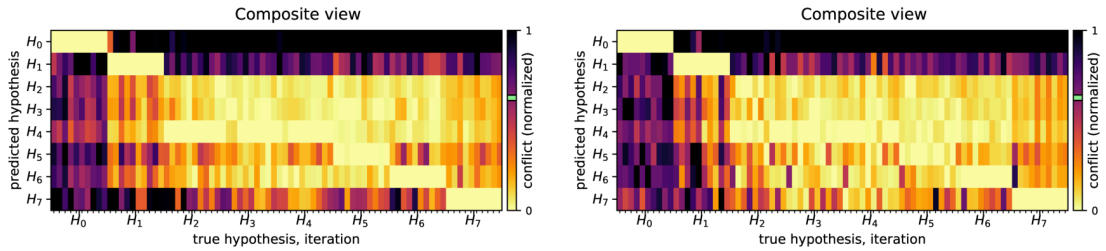
Stepsize	Pattern	$\Delta[\Xi]/\sum I$		Accuracy
-	-	μ	σ	-
0.5	$C[S]$	0.668	0.353	0.95
0.5	$C[SA]$	0.671	0.354	0.93
0.5	$C[R]$	0.748	0.332	0.47
0.5	$C[RA]$	0.766	0.343	0.64
1.0	$C[S]$	0.536	0.350	1.00
1.0	$C[SA]$	0.553	0.345	1.00
1.0	$C[R]$	0.650	0.366	0.68
1.0	$C[RA]$	0.666	0.355	0.74
1.5	$C[S]$	0.445	0.345	1.00
1.5	$C[SA]$	0.498	0.331	1.00
1.5	$C[R]$	0.609	0.378	0.70
1.5	$C[RA]$	0.622	0.355	0.72

Table D.13: Pattern and step size evaluation on $H^{\ominus}SE(3)$ utilizing Δ_{NW} with various patterns and step sizes. Intrinsic conflict (μ and σ) and classification accuracy.

Stepsize	Pattern	$\Lambda[\Xi]$		$\mathcal{S}$		$\mathcal{B}$		T_{CPU}		T_{WALL}		$\mathcal{S}/\Lambda[\Xi]$		$\mathcal{B}/\Lambda[\Xi]$	
-	-	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
0.5	$C[S]$	139.7	7.6	1865.5	29.0	1893.1	32.4	127.111	6.378	2.386	0.127	13.4	0.9	13.6	0.9
0.5	$C[SA]$	126.3	7.9	1843.6	27.9	1856.6	24.1	117.392	3.196	2.286	0.049	14.7	1.1	14.8	1.0
0.5	$C[R]$	105.2	4.5	4092.3	72.0	4092.3	72.0	363.691	19.697	6.634	0.265	39.0	2.2	39.0	2.2
0.5	$C[RA]$	86.4	3.6	3901.6	37.6	3901.6	37.6	379.751	15.858	6.750	0.181	45.2	1.8	45.2	1.8
1.0	$C[S]$	111.0	8.0	2018.7	38.8	2119.4	51.6	218.612	9.291	3.911	0.185	18.3	1.6	19.2	1.7
1.0	$C[SA]$	104.5	5.5	1970.3	38.2	2025.5	31.5	201.994	7.717	3.646	0.156	18.9	1.3	19.4	1.3
1.0	$C[R]$	85.3	2.7	4307.9	54.7	4324.1	56.0	536.581	17.596	9.025	0.256	50.5	1.6	50.7	1.6
1.0	$C[RA]$	70.9	3.1	4257.6	66.4	4269.0	68.2	573.096	37.698	9.451	0.531	60.2	3.2	60.3	3.3
1.5	$C[S]$	89.2	5.6	2073.2	33.3	2249.8	30.8	371.499	14.943	6.040	0.190	23.3	1.8	25.3	1.8
1.5	$C[SA]$	86.3	5.3	2019.3	33.7	2120.2	42.4	331.913	17.382	5.491	0.266	23.5	1.6	24.7	1.7
1.5	$C[R]$	70.3	1.7	4421.6	68.5	4473.2	67.0	1178.527	58.045	16.767	0.741	62.9	2.2	63.7	2.3
1.5	$C[RA]$	57.4	2.0	4465.6	55.6	4500.5	54.1	1338.510	26.057	18.788	0.270	77.9	2.7	78.5	2.8

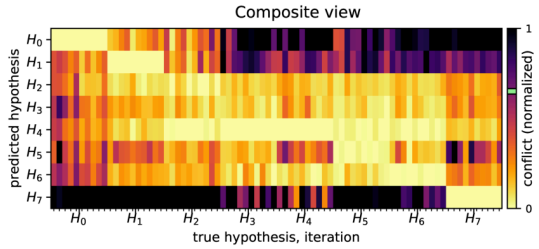
Table D.14: Pattern and step size evaluation on $H^{\ominus}SE(3)$ utilizing Δ_{NW} with various patterns and step sizes. Sensorimotor map properties and timing data.

D.5.2 Pattern and Step size Evaluation on H$^{\ominus}$SE(3) utilizing Δ_{RBFN}

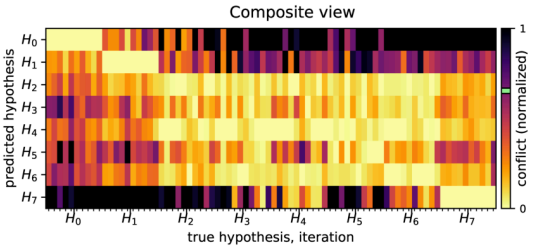


(a) Evaluation using pattern $C[S]$.

(b) Evaluation using pattern $C[SA]$.

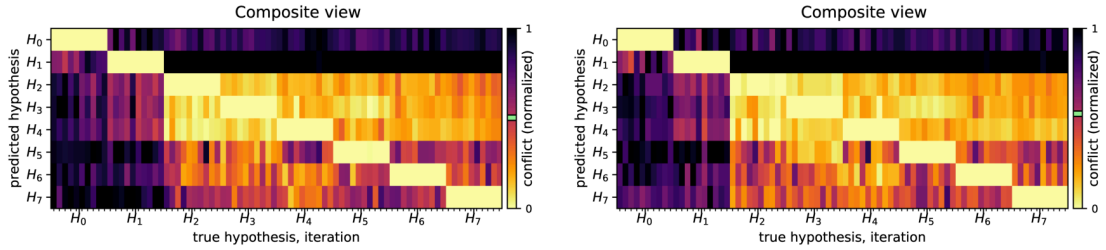


(c) Evaluation using pattern $C[R]$.



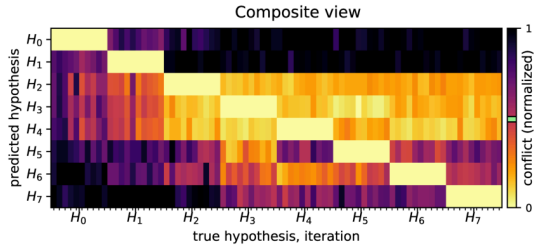
(d) Evaluation using pattern $C[RA]$.

Figure D.36: Pattern and step size evaluation on H$^{\ominus}$SE(3) utilizing Δ_{RBFN} with a step size of 0.5.

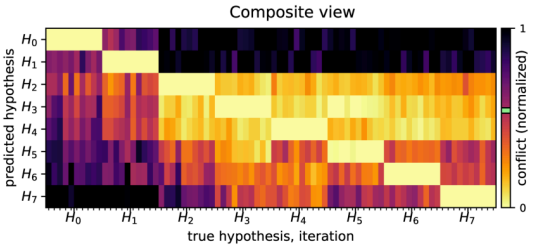


(a) Evaluation using pattern $C[S]$.

(b) Evaluation using pattern $C[SA]$.

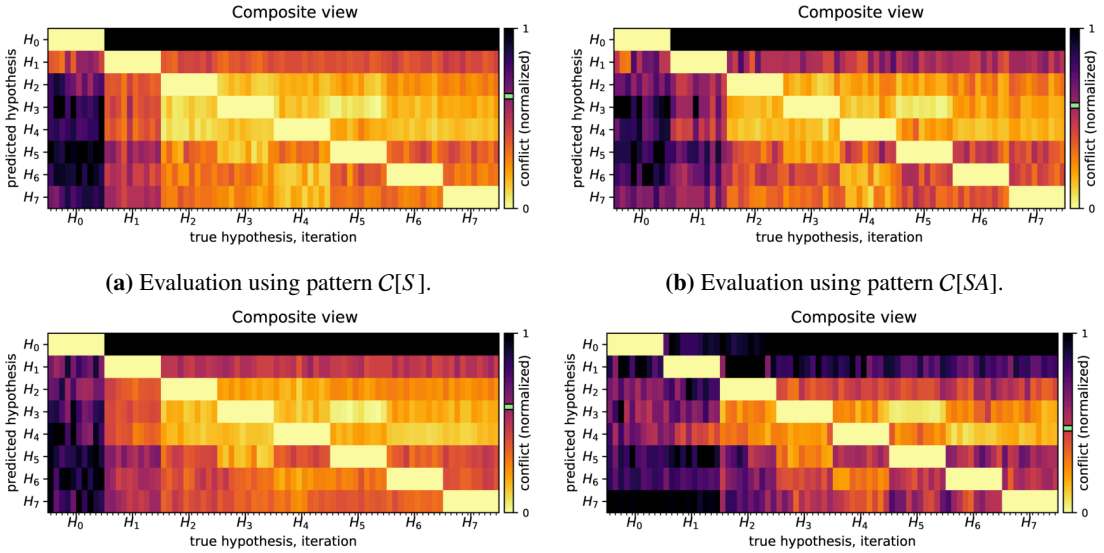


(c) Evaluation using pattern $C[R]$.



(d) Evaluation using pattern $C[RA]$.

Figure D.37: Pattern and step size evaluation on H$^{\ominus}$SE(3) utilizing Δ_{RBFN} with a step size of 1.0.



(a) Evaluation using pattern $C[S]$.

(b) Evaluation using pattern $C[SA]$.

(c) Evaluation using pattern $C[R]$.

(d) Evaluation using pattern $C[RA]$.

Figure D.38: Pattern and step size evaluation on H$^{\ominus}$SE(3) utilizing Δ_{RBFN} with a step size of 1.5.

Stepsize	Pattern	$\Delta[\Xi]/\sum I$		Accuracy
-	-	μ	σ	-
0.5	$C[S]$	0.625	0.364	0.74
0.5	$C[SA]$	0.626	0.364	0.68
0.5	$C[R]$	0.650	0.347	0.68
0.5	$C[RA]$	0.654	0.338	0.62
1.0	$C[S]$	0.504	0.358	0.99
1.0	$C[SA]$	0.526	0.360	0.96
1.0	$C[R]$	0.493	0.363	0.99
1.0	$C[RA]$	0.542	0.356	0.95
1.5	$C[S]$	0.623	0.320	1.00
1.5	$C[SA]$	0.571	0.317	1.00
1.5	$C[R]$	0.590	0.314	1.00
1.5	$C[RA]$	0.470	0.329	1.00

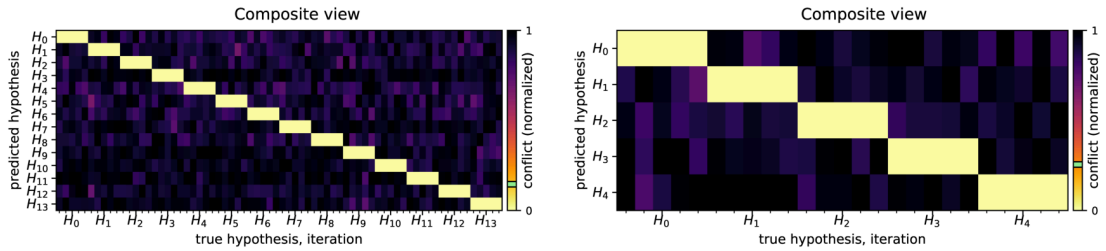
Table D.15: Pattern and step size evaluation on H$^{\ominus}$SE(3) utilizing Δ_{RBFN} with various patterns and step sizes. Intrinsic conflict (μ and σ) and classification accuracy.

Stepsize	Pattern	$\Lambda[\Xi]$		$\mathcal{S}$		$\mathcal{B}$		T_{CPU}		T_{WALL}		$\mathcal{S}/\Lambda[\Xi]$		$\mathcal{B}/\Lambda[\Xi]$	
-	-	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
0.5	$C[S]$	139.7	7.6	1865.5	29.0	1893.1	32.4	128.765	6.287	2.115	0.115	13.4	0.9	13.6	0.9
0.5	$C[SA]$	126.3	7.9	1843.6	27.9	1856.6	24.1	118.983	5.652	2.012	0.068	14.7	1.1	14.8	1.0
0.5	$C[R]$	105.2	4.5	4092.3	72.0	4092.3	72.0	278.183	9.532	6.087	0.274	39.0	2.2	39.0	2.2
0.5	$C[RA]$	86.4	3.6	3901.6	37.6	3901.6	37.6	277.253	7.927	6.166	0.158	45.2	1.8	45.2	1.8
1.0	$C[S]$	111.0	8.0	2018.7	38.8	2119.4	51.6	171.440	6.123	3.535	0.136	18.3	1.6	19.2	1.7
1.0	$C[SA]$	104.5	5.5	1970.3	38.2	2025.5	31.5	159.199	6.024	3.293	0.130	18.9	1.3	19.4	1.3
1.0	$C[R]$	85.3	2.7	4307.9	54.7	4324.1	56.0	372.087	11.988	8.421	0.284	50.5	1.6	50.7	1.6
1.0	$C[RA]$	70.9	3.1	4257.6	66.4	4269.0	68.2	389.207	20.621	8.917	0.487	60.2	3.2	60.3	3.3
1.5	$C[S]$	89.2	5.6	2073.2	33.3	2249.8	30.8	261.998	9.692	5.422	0.183	23.3	1.8	25.3	1.8
1.5	$C[SA]$	86.3	5.3	2019.3	33.7	2120.2	42.4	237.862	10.541	4.914	0.239	23.5	1.6	24.7	1.7
1.5	$C[R]$	70.3	1.7	4421.6	68.5	4473.2	67.0	747.907	34.704	15.183	0.701	62.9	2.2	63.7	2.3
1.5	$C[RA]$	57.4	2.0	4465.6	55.6	4500.5	54.1	849.476	16.333	17.523	0.324	77.9	2.7	78.5	2.8

Table D.16: Pattern and step size evaluation on H$^{\ominus}$SE(3) utilizing Δ_{RBFN} with various patterns and step sizes. Sensorimotor map properties and timing data.

D.6 Scenario IVa : Holonomic Baseline on SE(2)

D.6.1 Non-Noisy Actuator and Sensor Evaluation



(a) Evaluation on $H_1^{\oplus}SE(2)$.

(b) Evaluation on $H_2^{\oplus}SE(2)$.

Figure D.39: Non-noisy actuator and sensor evaluation on $H_1^{\oplus}SE(2)$ and $H_2^{\oplus}SE(2)$.

Noise	$\Delta[\Xi]/\sum I$		Accuracy
σ_c, σ_p	μ	σ	-
0.000	0.145	0.245	1.00

(a) Evaluation on $H_1^{\oplus}SE(2)$.

Noise	$\Delta[\Xi]/\sum I$		Accuracy
σ_c, σ_p	μ	σ	-
0.000	0.255	0.377	1.00

(b) Evaluation on $H_2^{\oplus}SE(2)$.

Table D.17: Non-noisy actuator and sensor evaluation on $H_1^{\oplus}SE(2)$ and $H_2^{\oplus}SE(2)$. Intrinsic conflict (μ and σ) and classification accuracy.

D.6.2 Sensor Noise Evaluation on $H_1^{\oplus}SE(2)$ and $H_2^{\oplus}SE(2)$

Noise	$\Delta[\Xi]/\sum I$		Accuracy
σ_p	μ	σ	-
0.001	0.145	0.245	1.00
0.003	0.145	0.245	1.00
0.005	0.145	0.245	1.00
0.010	0.145	0.245	1.00
0.025	0.145	0.245	1.00
0.050	0.145	0.245	1.00
0.100	0.145	0.244	1.00
0.250	0.145	0.244	1.00
0.500	0.147	0.244	1.00
0.750	0.151	0.243	1.00
1.000	0.155	0.242	1.00

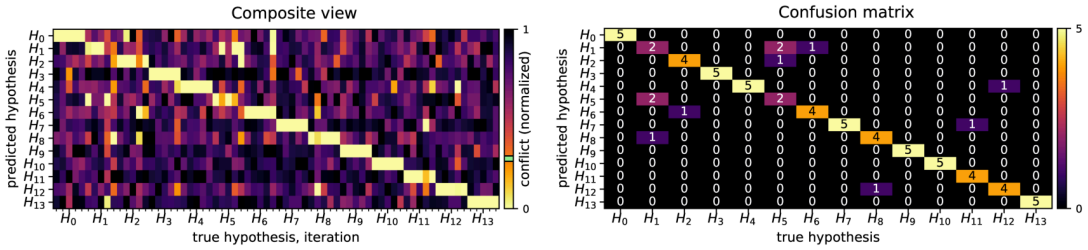
(a) Evaluation on $H_1^{\oplus}SE(2)$.

Noise	$\Delta[\Xi]/\sum I$		Accuracy
σ_p	μ	σ	-
0.001	0.255	0.377	1.00
0.003	0.255	0.377	1.00
0.005	0.255	0.377	1.00
0.010	0.255	0.377	1.00
0.025	0.255	0.377	1.00
0.050	0.255	0.377	1.00
0.100	0.255	0.377	1.00
0.250	0.254	0.378	1.00
0.500	0.253	0.378	1.00
0.750	0.255	0.378	1.00
1.000	0.258	0.377	1.00

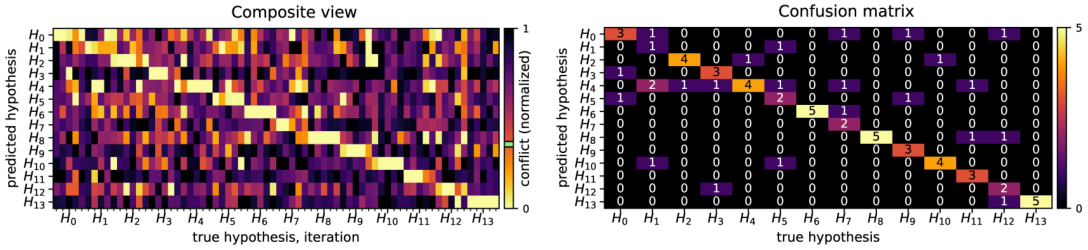
(b) Evaluation on $H_2^{\oplus}SE(2)$.

Table D.18: Sensor noise evaluation on $H_1^{\oplus}SE(2)$ and $H_2^{\oplus}SE(2)$. Intrinsic conflict (μ and σ) and classification accuracy.

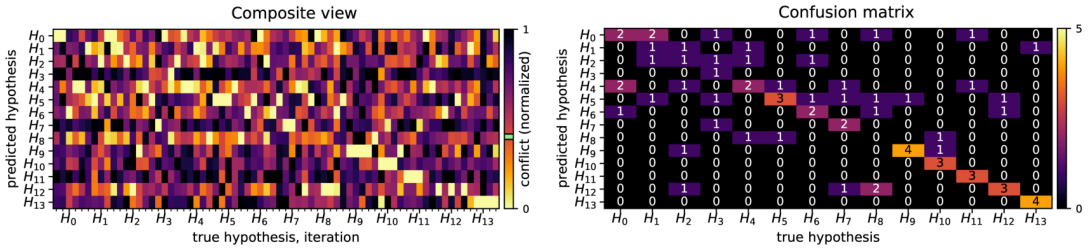
D.6.3 Actuator Noise Evaluation on $H_1^{\oplus}SE(2)$ and $H_2^{\oplus}SE(2)$



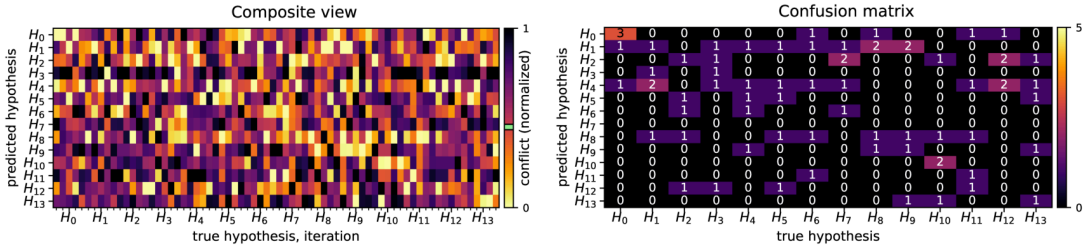
(a) Noisy actuator with σ_c set to 0.0250.



(b) Noisy actuator with σ_c set to 0.0500.

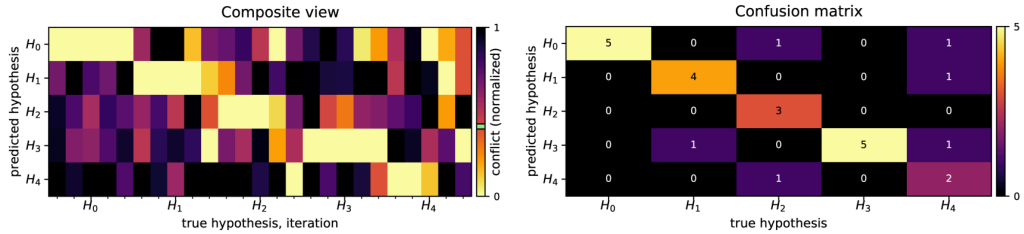


(c) Noisy actuator with σ_c set to 0.1000.

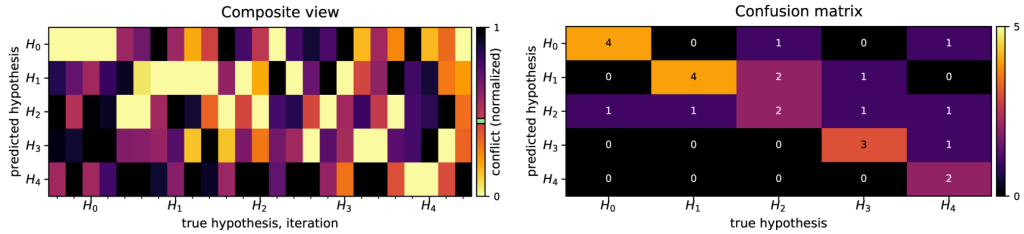


(d) Noisy actuator with σ_c set to 0.2500.

Figure D.40: Actuator noise evaluation on $H_1^{\oplus}SE(2)$ for $\sigma_c = 0.0250$ to $\sigma_c = 0.2500$.



(a) Noisy actuator with σ_c set to 0.0500.



(b) Noisy actuator with σ_c set to 0.1000.

Figure D.41: Actuator noise evaluation on $\mathrm{H}_2^{\oplus}\mathrm{SE}(2)$ for $\sigma_c = 0.0500$ and $\sigma_c = 0.1000$.

Noise	$\Delta[\Xi]/\sum I$		Accuracy
σ_c	μ	σ	-
0.001	0.156	0.244	1.00
0.003	0.160	0.243	1.00
0.005	0.174	0.243	1.00
0.010	0.197	0.250	0.99
0.025	0.279	0.272	0.84
0.050	0.357	0.296	0.66
0.100	0.401	0.295	0.44
0.250	0.449	0.307	0.20
0.500	0.434	0.303	0.21
0.750	0.424	0.302	0.07
1.000	0.428	0.304	0.07

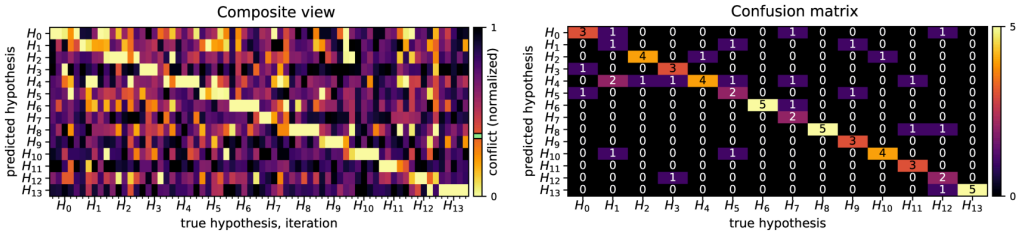
(a) Evaluation on $\mathrm{H}_1^{\oplus}\mathrm{SE}(2)$.

Noise	$\Delta[\Xi]/\sum I$		Accuracy
σ_c	μ	σ	-
0.001	0.266	0.374	1.00
0.003	0.263	0.377	1.00
0.005	0.262	0.377	1.00
0.010	0.276	0.373	1.00
0.025	0.340	0.359	1.00
0.050	0.413	0.379	0.76
0.100	0.446	0.369	0.60
0.250	0.481	0.382	0.16
0.500	0.461	0.377	0.32
0.750	0.482	0.385	0.24
1.000	0.484	0.379	0.24

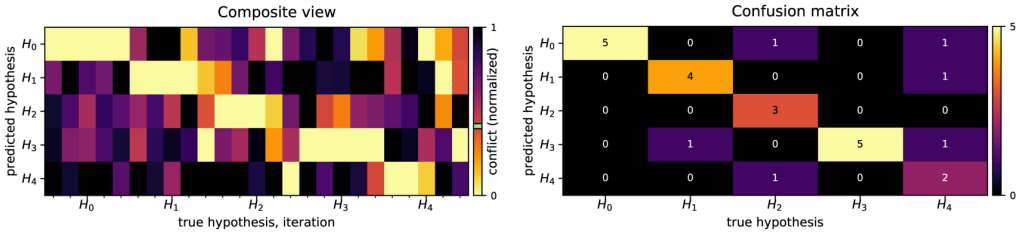
(b) Evaluation on $\mathrm{H}_2^{\oplus}\mathrm{SE}(2)$.

Table D.19: Actuator noise evaluation on $\mathrm{H}_1^{\oplus}\mathrm{SE}(2)$ and $\mathrm{H}_2^{\oplus}\mathrm{SE}(2)$. Intrinsic conflict ($\mu$ and σ) and classification accuracy.

D.6.4 Actuator and Sensor Noise Evaluation on $H_1^{\oplus}SE(2)$ and $H_2^{\oplus}SE(2)$



(a) Noisy actuator and sensors on $H_1^{\oplus}SE(2)$ with σ_c and σ_p set to 0.0500.



(b) Noisy actuator and sensors on $H_2^{\oplus}SE(2)$ with σ_c and σ_p set to 0.0500.

Figure D.42: Actuator and sensor noise evaluation on $H_1^{\oplus}SE(2)$ and $H_2^{\oplus}SE(2)$ for $\sigma_c, \sigma_p = 0.0500$.

Noise	$\Delta[\Xi]/\sum I$		Accuracy
σ_c, σ_p	μ	σ	-
0.001	0.156	0.244	1.00
0.003	0.160	0.243	1.00
0.005	0.174	0.243	1.00
0.010	0.197	0.250	0.99
0.025	0.278	0.272	0.84
0.050	0.357	0.296	0.66
0.100	0.402	0.295	0.43
0.250	0.445	0.307	0.20
0.500	0.437	0.305	0.17
0.750	0.417	0.298	0.10
1.000	0.448	0.299	0.06

(a) Evaluation on $H_1^{\oplus}SE(2)$.

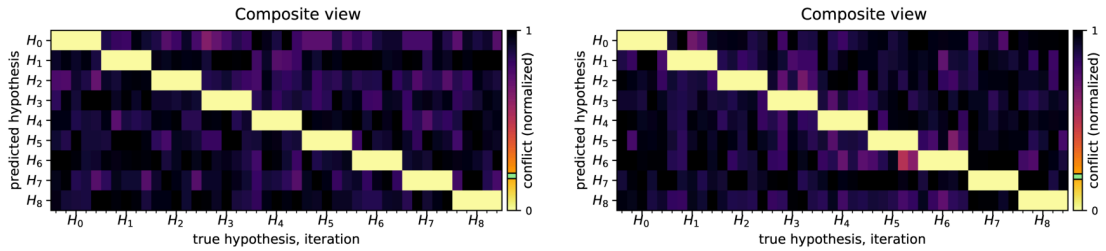
Noise	$\Delta[\Xi]/\sum I$		Accuracy
σ_c, σ_p	μ	σ	-
0.001	0.266	0.374	1.00
0.003	0.263	0.377	1.00
0.005	0.261	0.377	1.00
0.010	0.277	0.373	1.00
0.025	0.340	0.359	1.00
0.050	0.412	0.377	0.76
0.100	0.448	0.372	0.60
0.250	0.475	0.380	0.28
0.500	0.459	0.376	0.32
0.750	0.469	0.362	0.20
1.000	0.484	0.381	0.24

(b) Evaluation on $H_2^{\oplus}SE(2)$.

Table D.20: Actuator and sensor noise evaluation on $H_1^{\oplus}SE(2)$ and $H_2^{\oplus}SE(2)$. Intrinsic conflict (μ and σ) and classification accuracy.

D.7 Scenario IVb : Nonholonomic Baseline on SE(2)

D.7.1 Non-Noisy Actuator and Sensor Evaluation



(a) Evaluation on $H_1^{\ominus}SE(2)$. **(b)** Evaluation on $H_2^{\ominus}SE(2)$.

Figure D.43: Non-noisy actuator and sensor evaluation on $H_1^{\ominus}SE(2)$ and $H_2^{\ominus}SE(2)$.

Noise	$\Delta[\Xi]/\sum I$		Accuracy
σ_c, σ_p	μ	σ	-
0.000	0.192	0.294	1.00

(a) Evaluation on $H_1^{\ominus}SE(2)$.

Noise	$\Delta[\Xi]/\sum I$		Accuracy
σ_c, σ_p	μ	σ	-
0.000	0.188	0.296	1.00

(b) Evaluation on $H_2^{\ominus}SE(2)$.

Table D.21: Non-noisy actuator and sensor evaluation on $H_1^{\ominus}SE(2)$ and $H_2^{\ominus}SE(2)$. Intrinsic conflict (μ and σ) and classification accuracy.

D.7.2 Sensor Noise Evaluation on $H_1^{\ominus}SE(2)$ and $H_2^{\ominus}SE(2)$

Noise	$\Delta[\Xi]/\sum I$		Accuracy
σ_p	μ	σ	-
0.001	0.192	0.294	1.00
0.003	0.192	0.294	1.00
0.005	0.192	0.294	1.00
0.010	0.192	0.294	1.00
0.025	0.192	0.294	1.00
0.050	0.192	0.294	1.00
0.100	0.192	0.294	1.00
0.250	0.192	0.294	1.00
0.500	0.194	0.294	1.00
0.750	0.197	0.293	1.00
1.000	0.201	0.292	1.00

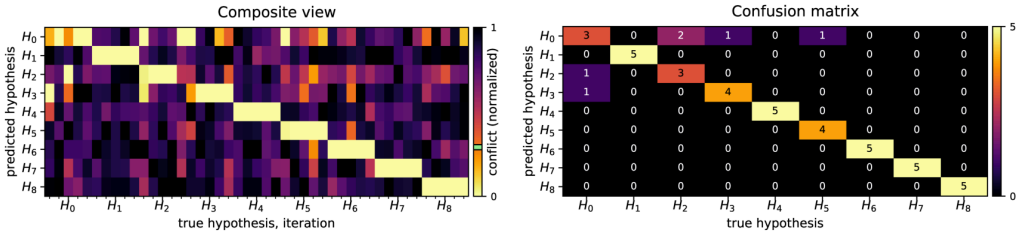
(a) Evaluation on $H_1^{\ominus}SE(2)$.

Noise	$\Delta[\Xi]/\sum I$		Accuracy
σ_p	μ	σ	-
0.001	0.188	0.296	1.00
0.003	0.188	0.296	1.00
0.005	0.188	0.296	1.00
0.010	0.188	0.296	1.00
0.025	0.188	0.296	1.00
0.050	0.188	0.296	1.00
0.100	0.188	0.296	1.00
0.250	0.189	0.296	1.00
0.500	0.191	0.295	1.00
0.750	0.195	0.294	1.00
1.000	0.200	0.293	1.00

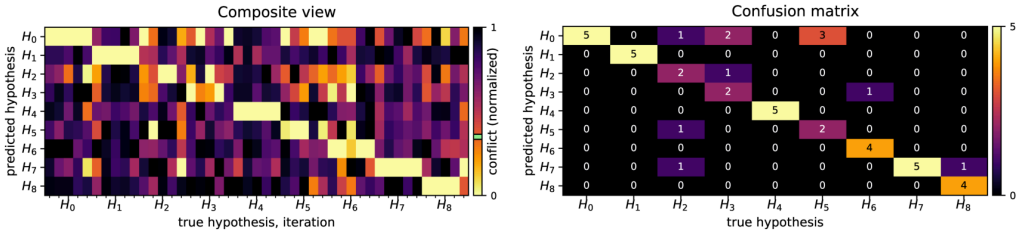
(b) Evaluation on $H_2^{\ominus}SE(2)$.

Table D.22: Sensor noise evaluation on $H_1^{\ominus}SE(2)$ and $H_2^{\ominus}SE(2)$. Intrinsic conflict (μ and σ) and classification accuracy.

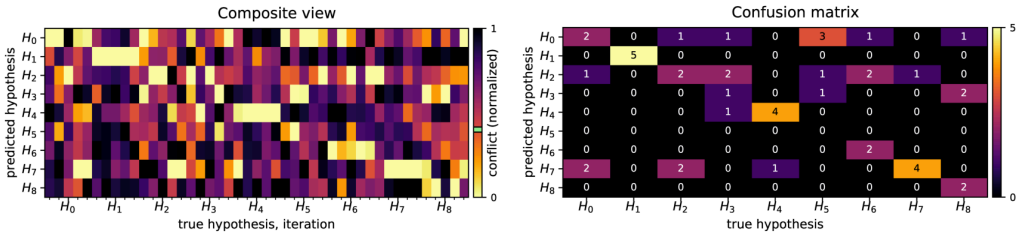
D.7.3 Actuator Noise Evaluation on $H_1^{\ominus}SE(2)$ and $H_2^{\ominus}SE(2)$



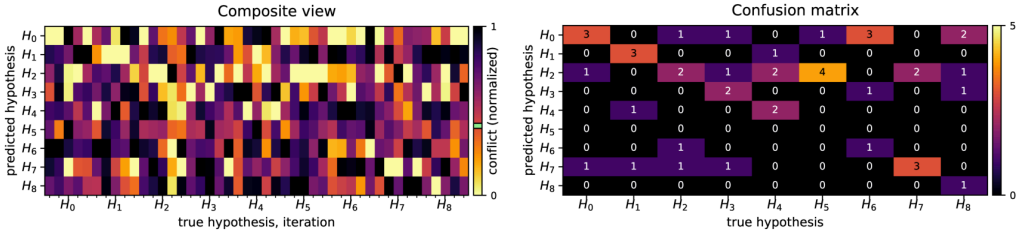
(a) Noisy actuator with σ_c set to 0.0250.



(b) Noisy actuator with σ_c set to 0.0500.

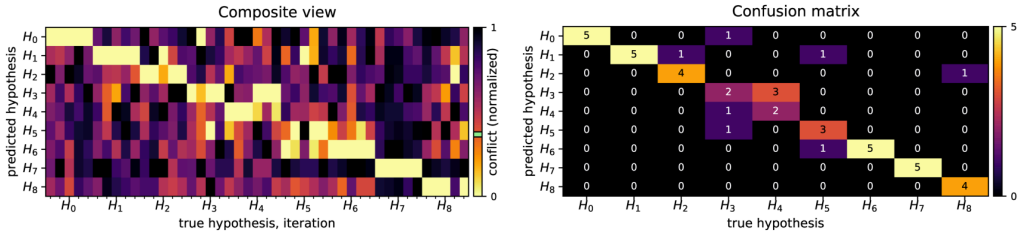


(c) Noisy actuator with σ_c set to 0.1000.

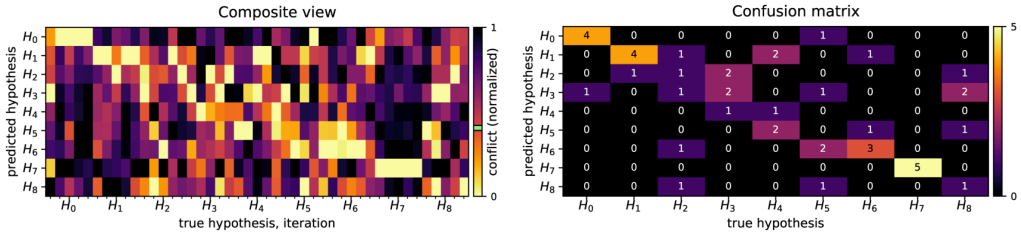


(d) Noisy actuator with σ_c set to 0.2500.

Figure D.44: Actuator noise evaluation on $H_1^{\ominus}SE(2)$ for $\sigma_c = 0.0250$ to $\sigma_c = 0.2500$.



(a) Noisy actuator with σ_c set to 0.0500.



(b) Noisy actuator with σ_c set to 0.1000.

Figure D.45: Actuator noise evaluation on $\mathrm{H}_2^{\ominus}\mathrm{SE}(2)$ for $\sigma_c = 0.0500$ and $\sigma_c = 0.1000$.

Noise	$\Delta[\Xi]/\sum I$		Accuracy
σ_c	μ	σ	-
0.001	0.197	0.293	1.00
0.003	0.204	0.294	1.00
0.005	0.213	0.296	1.00
0.010	0.233	0.295	1.00
0.025	0.290	0.311	0.87
0.050	0.349	0.324	0.76
0.100	0.401	0.325	0.49
0.250	0.411	0.327	0.38
0.500	0.426	0.336	0.16
0.750	0.445	0.328	0.11
1.000	0.412	0.327	0.09

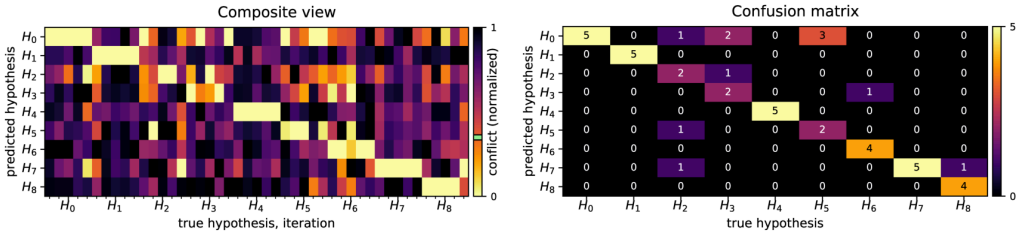
(a) Evaluation on $\mathrm{H}_1^{\ominus}\mathrm{SE}(2)$.

Noise	$\Delta[\Xi]/\sum I$		Accuracy
σ_c	μ	σ	-
0.001	0.194	0.295	1.00
0.003	0.203	0.294	1.00
0.005	0.211	0.294	1.00
0.010	0.233	0.295	1.00
0.025	0.312	0.315	0.93
0.050	0.366	0.313	0.78
0.100	0.405	0.329	0.47
0.250	0.443	0.328	0.29
0.500	0.458	0.322	0.16
0.750	0.467	0.332	0.11
1.000	0.454	0.328	0.11

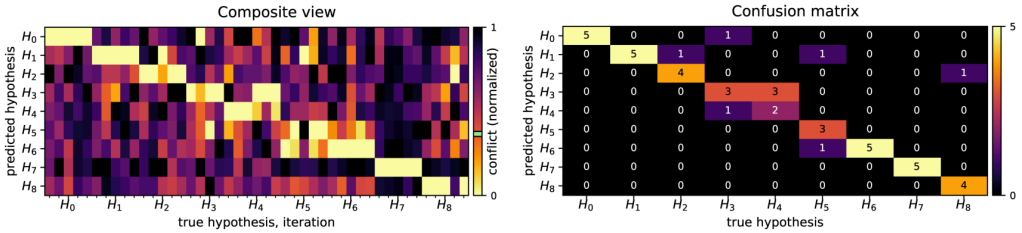
(b) Evaluation on $\mathrm{H}_2^{\ominus}\mathrm{SE}(2)$.

Table D.23: Actuator noise evaluation on $\mathrm{H}_1^{\ominus}\mathrm{SE}(2)$ and $\mathrm{H}_2^{\ominus}\mathrm{SE}(2)$. Intrinsic conflict ($\mu$ and σ) and classification accuracy.

D.7.4 Actuator and Sensor Noise Evaluation on $H_1^{\ominus}SE(2)$ and $H_2^{\ominus}SE(2)$



(a) Noisy actuator and sensors on $H_1^{\ominus}SE(2)$ with σ_c and σ_p set to 0.0500.



(b) Noisy actuator and sensors on $H_2^{\ominus}SE(2)$ with σ_c and σ_p set to 0.0500.

Figure D.46: Actuator and sensor noise evaluation on $H_1^{\oplus}SE(2)$ and $H_2^{\oplus}SE(2)$ for $\sigma_c, \sigma_p = 0.0500$.

Noise	$\Delta[\Xi]/\sum I$		Accuracy
σ_c, σ_p	μ	σ	-
0.001	0.197	0.293	1.00
0.003	0.204	0.294	1.00
0.005	0.213	0.296	1.00
0.010	0.233	0.295	1.00
0.025	0.290	0.311	0.87
0.050	0.349	0.324	0.76
0.100	0.401	0.326	0.51
0.250	0.417	0.327	0.36
0.500	0.431	0.334	0.18
0.750	0.451	0.336	0.16
1.000	0.432	0.331	0.09

(a) Evaluation on $H_1^{\ominus}SE(2)$.

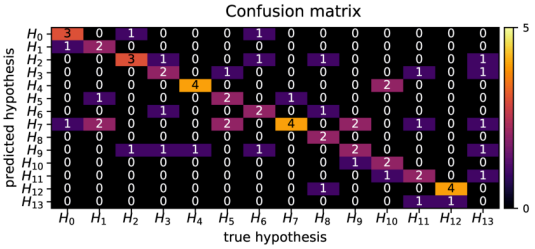
Noise	$\Delta[\Xi]/\sum I$		Accuracy
σ_c, σ_p	μ	σ	-
0.001	0.194	0.295	1.00
0.003	0.203	0.294	1.00
0.005	0.211	0.294	1.00
0.010	0.233	0.295	1.00
0.025	0.312	0.315	0.93
0.050	0.366	0.312	0.80
0.100	0.406	0.329	0.49
0.250	0.441	0.324	0.29
0.500	0.466	0.322	0.16
0.750	0.454	0.323	0.11
1.000	0.469	0.327	0.09

(b) Evaluation on $H_2^{\ominus}SE(2)$.

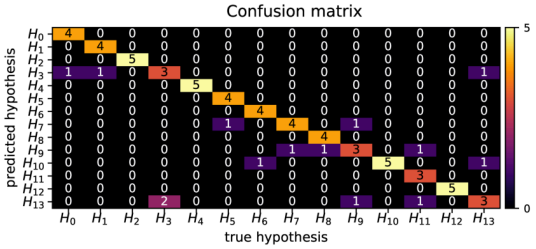
Table D.24: Actuator and sensor noise evaluation on $H_1^{\ominus}SE(2)$ and $H_2^{\ominus}SE(2)$. Intrinsic conflict (μ and σ) and classification accuracy.

D.8 Scenario Va : Holonomic Classification on SE(2)

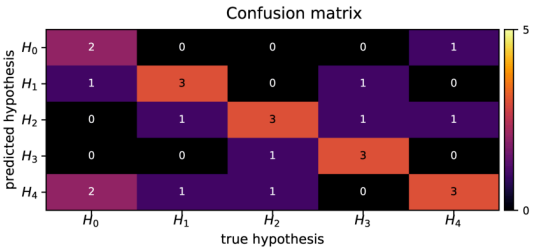
D.8.1 Intermediate Conflict Calculation on $H_1^{\oplus}SE(2)$ and $H_2^{\oplus}SE(2)$



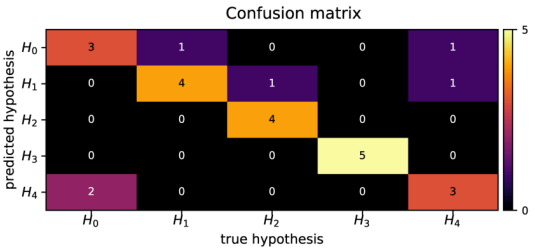
(a) Evaluation on $H_1^{\oplus}SE(2)$ for $t = 50$.



(b) Evaluation on $H_1^{\oplus}SE(2)$ for $t = 100$.



(c) Evaluation on $H_2^{\oplus}SE(2)$ for $t = 50$.



(d) Evaluation on $H_2^{\oplus}SE(2)$ for $t = 100$.

Figure D.47: Intermediate conflict calculation on $H_1^{\oplus}SE(2)$ and $H_2^{\oplus}SE(2)$,for $t = 50$ and $t = 100$.

Timestep	$\Delta[\Xi]/\sum I$		Accuracy
-	μ	σ	-
50	0.562	0.301	0.49
100	0.496	0.289	0.80
150	0.458	0.277	0.91
200	0.410	0.264	0.99
250	0.376	0.256	1.00
300	0.361	0.252	1.00
350	0.344	0.253	1.00
400	0.332	0.252	1.00
450	0.323	0.252	1.00
500	0.314	0.251	1.00
550	0.304	0.251	1.00
600	0.302	0.248	1.00
650	0.294	0.245	1.00
700	0.289	0.245	1.00
750	0.284	0.244	1.00
800	0.277	0.243	1.00
850	0.271	0.243	1.00
900	0.263	0.243	1.00
950	0.261	0.243	1.00
1000	0.259	0.242	1.00

(a) Evaluation on $H_1^{\oplus}SE(2)$.

Timestep	$\Delta[\Xi]/\sum I$		Accuracy
-	μ	σ	-
50	0.533	0.387	0.56
100	0.503	0.387	0.76
150	0.458	0.377	0.92
200	0.420	0.379	0.96
250	0.411	0.383	1.00
300	0.416	0.372	1.00
350	0.393	0.373	1.00
400	0.384	0.364	1.00
450	0.375	0.364	1.00
500	0.369	0.359	1.00
550	0.357	0.361	1.00
600	0.353	0.362	1.00
650	0.349	0.360	1.00
700	0.341	0.360	1.00
750	0.332	0.362	1.00
800	0.329	0.360	1.00
850	0.320	0.363	1.00
900	0.313	0.365	1.00
950	0.310	0.365	1.00
1000	0.311	0.364	1.00

(b) Evaluation on $H_2^{\oplus}SE(2)$.

Table D.25: Intermediate conflict calculation on $H_1^{\oplus}SE(2)$ and $H_2^{\oplus}SE(2)$ for $t = 50$ to $t = 1000$. Intrinsic conflict (μ and σ) and classification accuracy.

Timestep	$\Lambda[\Xi]$		$\mathcal{S}$		$\mathcal{B}$		T_{CPU}		T_{WALL}		$\mathcal{S}/\Lambda[\Xi]$		$\mathcal{B}/\Lambda[\Xi]$	
-	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
50	10.6	1.2	88.7	3.8	92.4	2.9	3.134	0.212	0.058	0.003	8.5	0.9	8.8	0.9
100	18.6	2.1	183.7	4.5	198.5	3.2	10.506	1.202	0.203	0.013	10.0	1.5	10.8	1.5
150	25.9	2.9	280.5	8.3	309.4	9.1	15.555	0.704	0.313	0.010	11.0	1.7	12.1	1.8
200	33.4	2.7	378.2	9.4	418.4	6.3	21.282	1.473	0.433	0.030	11.4	1.2	12.6	1.2
250	40.0	2.4	476.8	10.5	529.8	7.0	27.404	0.535	0.564	0.014	12.0	0.9	13.3	0.9
300	46.2	2.4	576.9	11.6	641.3	6.9	33.785	0.778	0.701	0.023	12.5	0.8	13.9	0.9
350	53.4	1.5	676.0	10.1	751.4	7.1	39.521	0.794	0.810	0.020	12.7	0.5	14.1	0.5
400	59.9	2.0	775.2	10.6	864.0	9.2	45.502	0.550	0.924	0.020	13.0	0.6	14.4	0.6
450	65.7	2.4	878.5	12.2	979.6	12.6	51.787	1.161	1.053	0.035	13.4	0.6	14.9	0.7
500	72.2	2.5	979.6	11.7	1093.5	12.0	58.935	1.058	1.186	0.033	13.6	0.6	15.2	0.7
550	78.4	2.6	1081.5	9.7	1207.2	11.7	66.029	1.636	1.327	0.041	13.8	0.6	15.4	0.7
600	85.1	2.3	1183.4	8.7	1321.5	12.5	72.942	2.646	1.464	0.064	13.9	0.5	15.5	0.6
650	91.8	2.5	1285.6	5.9	1438.9	14.3	81.628	2.659	1.646	0.061	14.0	0.4	15.7	0.6
700	98.3	2.5	1387.5	6.5	1552.1	14.2	88.671	2.564	1.789	0.055	14.1	0.4	15.8	0.6
750	104.3	3.7	1490.2	10.7	1667.8	15.2	96.705	2.423	1.954	0.057	14.3	0.6	16.0	0.7
800	110.7	3.5	1591.2	12.4	1783.0	17.5	105.130	2.945	2.123	0.074	14.4	0.6	16.1	0.7
850	116.6	3.2	1693.9	13.1	1898.9	17.7	112.717	2.746	2.267	0.072	14.5	0.5	16.3	0.6
900	122.7	3.5	1798.1	13.8	2015.7	15.2	121.315	1.940	2.430	0.059	14.7	0.5	16.4	0.6
950	129.4	3.7	1901.4	13.3	2131.0	15.2	130.371	2.168	2.589	0.069	14.7	0.5	16.5	0.6
1000	135.6	4.1	2002.8	16.2	2244.6	16.0	138.547	1.408	2.729	0.054	14.8	0.6	16.6	0.6

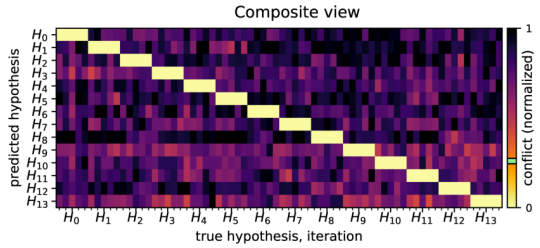
(a) Intermediate conflict calculation on $H_1^{\oplus}SE(2)$.

Timestep	$\Lambda[\Xi]$		$\mathcal{S}$		$\mathcal{B}$		T_{CPU}		T_{WALL}		$\mathcal{S}/\Lambda[\Xi]$		$\mathcal{B}/\Lambda[\Xi]$	
-	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
50	10.9	2.0	88.8	5.6	90.6	4.8	3.218	0.304	0.060	0.007	8.3	1.3	8.5	1.3
100	18.0	3.4	179.7	6.2	193.7	7.0	9.175	2.116	0.182	0.031	10.3	2.4	11.1	2.3
150	23.9	4.3	277.6	9.4	303.7	9.8	14.330	0.714	0.300	0.023	12.0	3.0	13.1	3.0
200	29.6	3.8	380.6	6.5	419.3	7.2	20.070	1.090	0.432	0.017	13.0	1.9	14.3	1.8
250	34.1	3.5	478.8	10.5	530.9	9.3	25.749	1.380	0.566	0.042	14.2	1.7	15.7	1.7
300	39.1	2.7	580.3	12.2	640.2	13.9	32.486	1.802	0.714	0.043	14.9	1.3	16.4	1.3
350	44.5	2.5	680.3	13.0	750.9	12.5	37.031	1.764	0.813	0.029	15.3	1.1	16.9	1.1
400	49.8	3.7	781.8	18.0	864.4	20.4	44.239	1.881	0.962	0.041	15.8	1.5	17.5	1.6
450	53.9	4.4	888.0	25.1	983.2	24.9	50.817	3.385	1.105	0.089	16.6	1.8	18.4	1.9
500	58.4	4.8	993.9	28.4	1103.0	29.6	58.041	3.285	1.246	0.091	17.1	1.9	19.0	2.0
550	62.4	5.2	1098.3	28.5	1225.6	30.1	65.848	3.855	1.395	0.107	17.7	1.9	19.8	2.1
600	66.8	5.2	1201.5	30.4	1342.4	32.8	74.043	6.524	1.556	0.149	18.1	1.9	20.2	2.0
650	71.5	5.3	1306.2	31.9	1464.8	36.9	83.602	7.050	1.748	0.161	18.4	1.8	20.6	2.0
700	75.9	5.9	1410.4	33.4	1582.3	36.3	91.750	6.666	1.906	0.133	18.7	1.9	21.0	2.1
750	80.7	6.9	1516.6	37.9	1701.0	40.2	98.972	6.519	2.055	0.124	18.9	2.2	21.2	2.4
800	85.6	6.3	1619.4	37.7	1819.7	45.7	107.645	9.053	2.226	0.180	19.0	1.9	21.4	2.1
850	89.9	5.5	1725.3	38.8	1941.9	43.7	116.967	9.940	2.427	0.193	19.3	1.6	21.7	1.7
900	93.9	4.6	1832.8	37.9	2062.5	36.8	126.074	6.789	2.607	0.130	19.6	1.3	22.0	1.4
950	97.9	4.5	1941.3	38.7	2187.4	29.5	135.295	7.772	2.793	0.154	19.9	1.3	22.4	1.3
1000	101.8	4.8	2046.9	41.7	2305.1	35.7	144.867	6.478	2.996	0.135	20.2	1.3	22.7	1.4

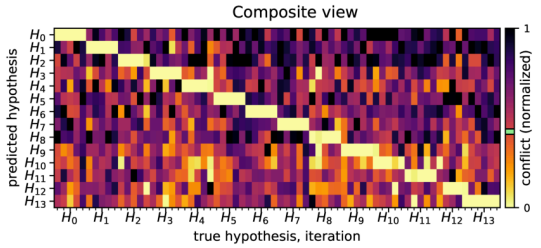
(b) Intermediate conflict calculation on $H_2^{\oplus}SE(2)$.

Table D.26: Intermediate conflict calculation on $H_1^{\oplus}SE(2)$ and $H_2^{\oplus}SE(2)$ for $t = 50$ to $t = 1000$. Sensorimotor map properties and timing data.

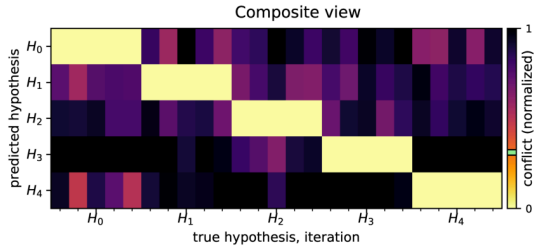
D.8.2 Subset Conflict Calculation on $H_1^{\oplus}SE(2)$ and $H_2^{\oplus}SE(2)$



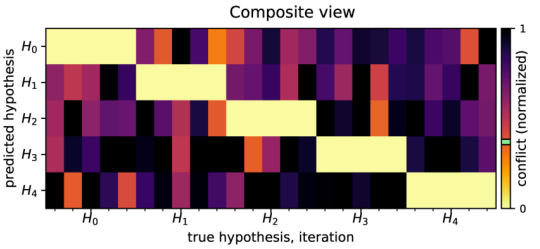
(a) Evaluation on $H_1^{\oplus}SE(2)$ for $n = 1$ and $r = 1.0$.



(b) Evaluation on $H_1^{\oplus}SE(2)$ for $n = 10$ and $r = 0.1$.



(c) Evaluation on $H_2^{\oplus}SE(2)$ for $n = 1$ and $r = 1.0$.



(d) Evaluation on $H_2^{\oplus}SE(2)$ for $n = 10$ and $r = 0.1$.

Figure D.48: Subset conflict calculation on $H_1^{\oplus}SE(2)$ and $H_2^{\oplus}SE(2)$, for $(n = 1, r = 1.0)$ and $(n = 10, r = 0.1)$.

Subsets	Ratio	$\Delta[\Xi]/\sum I$		Accuracy
n	r	μ	σ	-
1	1.000	0.259	0.242	1.00
2	0.500	0.245	0.241	1.00
3	0.333	0.241	0.243	1.00
4	0.250	0.255	0.243	1.00
5	0.200	0.270	0.245	1.00
6	0.166	0.308	0.250	1.00
7	0.142	0.342	0.255	0.99
8	0.125	0.366	0.260	0.99
9	0.111	0.404	0.274	0.96
10	0.100	0.424	0.280	0.90

(a) Subset conflict calculation for $H_1^{\oplus}SE(2)$. Intrinsic conflict (μ and σ) and classification accuracy.

Subsets	Ratio	T_{CPU}		T_{WALL}	
n	r	μ	σ	μ	σ
1	1.000	133.132	1.987	2.623	0.062
2	0.500	132.511	1.882	2.186	0.039
3	0.333	138.472	2.197	2.062	0.042
4	0.250	153.238	3.016	2.177	0.041
5	0.200	163.276	1.936	2.262	0.018
6	0.166	173.739	6.208	2.381	0.092
7	0.142	182.913	5.440	2.484	0.074
8	0.125	197.826	5.062	2.671	0.078
9	0.111	202.923	10.040	2.726	0.145
10	0.100	216.174	6.378	2.903	0.087

(b) Subset conflict calculation for $H_1^{\oplus}SE(2)$. Timing data.

Subsets	Ratio	$\Delta[\Xi]/\sum I$		Accuracy
n	r	μ	σ	-
1	1.000	0.311	0.364	1.00
2	0.500	0.320	0.363	1.00
3	0.333	0.320	0.365	1.00
4	0.250	0.320	0.365	1.00
5	0.200	0.312	0.368	1.00
6	0.166	0.311	0.369	1.00
7	0.142	0.325	0.365	1.00
8	0.125	0.342	0.363	1.00
9	0.111	0.361	0.365	1.00
10	0.100	0.369	0.365	1.00

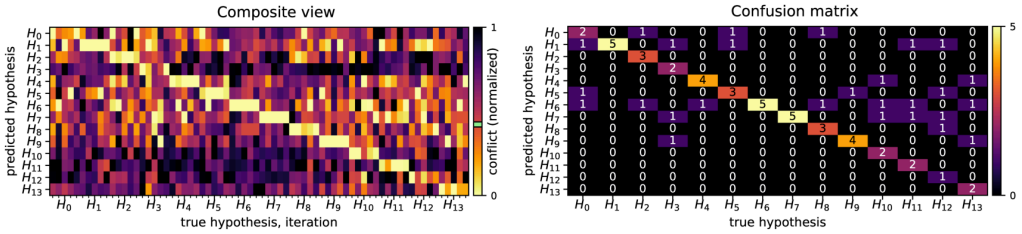
(c) Subset conflict calculation for $H_2^{\oplus}SE(2)$. Intrinsic conflict (μ and σ) and classification accuracy.

Subsets	Ratio	T_{CPU}		T_{WALL}	
n	r	μ	σ	μ	σ
1	1.000	140.806	5.755	2.906	0.120
2	0.500	120.689	3.460	2.225	0.075
3	0.333	122.904	2.251	2.010	0.054
4	0.250	133.838	5.959	2.037	0.096
5	0.200	140.438	4.775	2.037	0.061
6	0.166	143.787	6.462	2.012	0.088
7	0.142	153.953	6.723	2.120	0.087
8	0.125	162.840	13.045	2.229	0.186
9	0.111	161.614	14.986	2.180	0.224
10	0.100	173.990	6.522	2.348	0.097

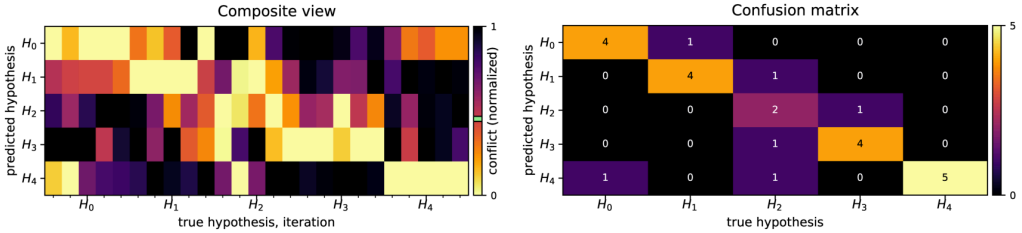
(d) Subset conflict calculation for $H_2^{\oplus}SE(2)$. Timing data.

Table D.27: Subset conflict calculation on $H_1^{\oplus}SE(2)$ and $H_2^{\oplus}SE(2)$ for $(n = 1, r = 1.0)$ to $(n = 10, r = 0.1)$. Intrinsic conflict (μ and σ) and classification accuracy. Timing data.

D.8.3 Sensor Noise Evaluation on $H_1^{\oplus}SE(2)$ and $H_2^{\oplus}SE(2)$



(a) Noisy sensors on $H_1^{\oplus}SE(2)$ with σ_p set to 1.000.



(b) Noisy sensors on $H_1^{\oplus}SE(2)$ with σ_p set to 1.000.

Figure D.49: Sensor noise evaluation on $H_1^{\oplus}SE(2)$ and $H_2^{\oplus}SE(2)$ for $\sigma_p = 1.000$.

Noise	$\Delta[\Xi]/\sum I$		Accuracy
σ_p	μ	σ	-
0.001	0.259	0.242	1.00
0.003	0.260	0.242	1.00
0.005	0.260	0.242	1.00
0.010	0.260	0.242	1.00
0.025	0.260	0.242	1.00
0.050	0.260	0.242	1.00
0.100	0.257	0.241	1.00
0.250	0.248	0.240	1.00
0.500	0.310	0.258	0.99
0.750	0.389	0.287	0.81
1.000	0.422	0.295	0.61

(a) Sensor noise evaluation for $H_1^{\oplus}SE(2)$. Intrinsic conflict (μ and σ) and classification accuracy.

Noise	T_{CPU}		T_{WALL}	
σ_p	μ	σ	μ	σ
0.001	133.208	1.592	2.621	0.052
0.003	131.729	2.233	2.590	0.061
0.005	130.821	1.814	2.575	0.051
0.010	131.963	1.512	2.594	0.045
0.025	130.970	2.507	2.571	0.067
0.050	131.414	2.360	2.585	0.063
0.100	131.728	2.020	2.591	0.062
0.250	131.111	2.604	2.575	0.065
0.500	131.242	1.606	2.580	0.052
0.750	131.431	2.145	2.582	0.057
1.000	131.115	1.995	2.577	0.054

(b) Sensor noise evaluation for $H_1^{\oplus}SE(2)$. Timing data.

Noise	$\Delta[\Xi]/\sum I$		Accuracy
σ_p	μ	σ	-
0.001	0.311	0.364	1.00
0.003	0.311	0.364	1.00
0.005	0.311	0.364	1.00
0.010	0.311	0.364	1.00
0.025	0.311	0.365	1.00
0.050	0.311	0.365	1.00
0.100	0.311	0.365	1.00
0.250	0.316	0.366	1.00
0.500	0.362	0.367	1.00
0.750	0.425	0.387	0.92
1.000	0.453	0.392	0.76

(c) Sensor noise evaluation for $H_2^{\oplus}SE(2)$. Intrinsic conflict (μ and σ) and classification accuracy.

Noise	T_{CPU}		T_{WALL}	
σ_p	μ	σ	μ	σ
0.001	140.773	6.267	2.914	0.119
0.003	141.457	8.240	2.933	0.172
0.005	140.828	6.371	2.912	0.132
0.010	141.548	6.688	2.925	0.140
0.025	141.332	6.720	2.923	0.139
0.050	140.614	5.647	2.908	0.117
0.100	140.882	6.763	2.915	0.130
0.250	139.623	6.400	2.897	0.125
0.500	140.518	6.601	2.910	0.133
0.750	140.694	5.320	2.901	0.104
1.000	140.914	7.018	2.911	0.129

(d) Sensor noise evaluation for $H_2^{\oplus}SE(2)$. Timing data.

Table D.28: Sensor noise evaluation on $H_1^{\oplus}SE(2)$ and $H_2^{\oplus}SE(2)$. Intrinsic conflict (μ and σ) and classification accuracy. Timing data.

D.8.4 Actuator Noise Evaluation on Hypothesis Set $H_1^{\oplus}SE(2)$ and $H_2^{\oplus}SE(2)$

Noise	$\Delta[\Xi]/\sum I$		Accuracy
σ_c	μ	σ	-
0.001	0.254	0.240	1.00
0.003	0.256	0.244	1.00
0.005	0.263	0.242	1.00
0.010	0.277	0.246	1.00
0.025	0.312	0.251	1.00
0.050	0.351	0.263	1.00
0.100	0.406	0.282	0.91
0.250	0.472	0.296	0.59
0.500	0.500	0.295	0.26
0.750	0.500	0.292	0.13
1.000	0.498	0.297	0.09

(a) Actuator noise evaluation on $H_1^{\oplus}SE(2)$.

Noise	$\Delta[\Xi]/\sum I$		Accuracy
σ_c	μ	σ	-
0.001	0.330	0.362	1.00
0.003	0.323	0.360	1.00
0.005	0.316	0.363	1.00
0.010	0.330	0.365	1.00
0.025	0.349	0.362	1.00
0.050	0.416	0.371	1.00
0.100	0.432	0.370	0.92
0.250	0.522	0.384	0.40
0.500	0.518	0.381	0.28
0.750	0.500	0.374	0.16
1.000	0.502	0.380	0.28

(b) Actuator noise evaluation on $H_2^{\oplus}SE(2)$.

Table D.29: Actuator noise evaluation on $H_1^{\oplus}SE(2)$ and $H_2^{\oplus}SE(2)$. Intrinsic conflict (μ and σ) and classification accuracy.

Noise	$\Lambda[\Xi]$		$\mathcal{S}$		$\mathcal{B}$		T_{CPU}		T_{WALL}		$\mathcal{S}/\Lambda[\Xi]$		$\mathcal{B}/\Lambda[\Xi]$	
σ_c	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
0.001	135.6	3.0	2005.1	10.8	2246.1	20.1	131.310	1.691	2.579	0.049	14.8	0.4	16.6	0.5
0.003	134.9	2.9	2008.6	16.9	2245.8	19.8	130.381	2.737	2.570	0.063	14.9	0.4	16.7	0.5
0.005	135.2	4.0	2004.8	15.2	2251.5	19.5	130.901	2.193	2.580	0.061	14.8	0.5	16.7	0.6
0.010	135.5	3.8	2000.8	19.0	2244.9	16.2	130.259	2.743	2.561	0.061	14.8	0.5	16.6	0.6
0.025	135.8	2.3	1991.4	17.0	2243.7	23.8	130.503	3.446	2.565	0.068	14.7	0.4	16.5	0.4
0.050	136.5	2.1	1998.6	20.3	2246.6	17.2	129.686	2.707	2.550	0.050	14.6	0.3	16.5	0.3
0.100	137.0	3.3	1996.1	23.9	2255.5	25.5	131.352	2.718	2.580	0.069	14.6	0.5	16.5	0.6
0.250	139.8	3.9	1989.6	19.2	2234.8	18.7	128.334	1.862	2.500	0.065	14.2	0.5	16.0	0.6
0.500	141.1	3.8	1982.6	16.4	2257.7	20.2	131.633	1.840	2.563	0.065	14.1	0.5	16.0	0.6
0.750	141.1	4.1	1977.1	16.8	2278.5	33.7	133.765	3.302	2.605	0.089	14.0	0.5	16.2	0.6
1.000	141.2	5.6	1976.9	27.2	2304.9	29.4	135.714	2.189	2.648	0.082	14.0	0.7	16.3	0.8

(a) Actuator noise evaluation on $H_1^{\oplus}SE(2)$.

Noise	$\Lambda[\Xi]$		$\mathcal{S}$		$\mathcal{B}$		T_{CPU}		T_{WALL}		$\mathcal{S}/\Lambda[\Xi]$		$\mathcal{B}/\Lambda[\Xi]$	
σ_c	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
0.001	103.6	8.1	2045.3	46.4	2315.6	48.0	143.495	14.507	2.959	0.260	19.9	2.1	22.5	2.3
0.003	104.6	5.1	2048.9	35.1	2297.5	50.6	138.730	6.705	2.875	0.151	19.6	1.2	22.0	1.5
0.005	103.1	7.5	2048.2	59.9	2323.8	68.1	143.462	13.359	2.971	0.285	20.0	1.9	22.7	2.2
0.010	105.9	4.1	2037.0	45.6	2308.0	16.6	139.218	5.402	2.889	0.110	19.3	1.0	21.8	0.9
0.025	106.4	8.0	2033.5	43.1	2296.9	26.2	136.569	4.487	2.827	0.089	19.2	1.5	21.7	1.7
0.050	109.5	6.8	2018.6	42.2	2254.4	33.0	130.322	6.172	2.715	0.139	18.5	1.4	20.6	1.4
0.100	113.9	5.9	2014.0	35.1	2275.8	44.9	134.403	4.550	2.771	0.102	17.7	1.2	20.0	1.1
0.250	119.9	8.2	1998.5	48.6	2252.4	57.1	133.153	8.346	2.733	0.191	16.8	1.5	18.9	1.8
0.500	124.0	10.4	1992.9	37.9	2278.1	89.4	132.743	11.509	2.694	0.289	16.2	1.6	18.5	2.3
0.750	128.0	6.0	1972.8	28.6	2283.1	47.4	132.776	4.107	2.696	0.125	15.4	0.9	17.9	1.0
1.000	126.6	10.2	1965.7	51.4	2315.8	44.6	136.167	2.712	2.778	0.130	15.6	1.7	18.4	1.8

(b) Actuator noise evaluation on $H_2^{\oplus}SE(2)$.

Table D.30: Actuator noise evaluation on $H_1^{\oplus}SE(2)$ and $H_2^{\oplus}SE(2)$. Sensorimotor map properties and timing data.

D.8.5 Actuator and Sensor Noise Evaluation on $H_1^{\oplus}SE(2)$ and $H_2^{\oplus}SE(2)$

Noise	$\Delta[\Xi]/\sum I$		Accuracy
σ_c, σ_p	μ	σ	-
0.001	0.254	0.240	1.00
0.003	0.256	0.244	1.00
0.005	0.263	0.242	1.00
0.010	0.277	0.246	1.00
0.025	0.311	0.251	1.00
0.050	0.349	0.263	1.00
0.100	0.394	0.279	0.94
0.250	0.473	0.294	0.64
0.500	0.514	0.290	0.24
0.750	0.489	0.290	0.14
1.000	0.489	0.299	0.13

(a) Actuator and sensor noise evaluation on $H_1^{\oplus}SE(2)$.

Noise	$\Delta[\Xi]/\sum I$		Accuracy
σ_c, σ_p	μ	σ	-
0.001	0.330	0.362	1.00
0.003	0.323	0.360	1.00
0.005	0.316	0.363	1.00
0.010	0.330	0.365	1.00
0.025	0.347	0.363	1.00
0.050	0.415	0.370	1.00
0.100	0.428	0.369	0.92
0.250	0.533	0.376	0.52
0.500	0.513	0.373	0.28
0.750	0.512	0.383	0.08
1.000	0.502	0.369	0.04

(b) Actuator and sensor noise evaluation on $H_2^{\oplus}SE(2)$.

Table D.31: Actuator and sensor noise evaluation on $H_1^{\oplus}SE(2)$ and $H_2^{\oplus}SE(2)$. Intrinsic conflict (μ and σ) and classification accuracy.

Noise	$\Lambda[\Xi]$		$\mathcal{S}$		$\mathcal{B}$		T_{CPU}		T_{WALL}		$\mathcal{S}/\Lambda[\Xi]$		$\mathcal{B}/\Lambda[\Xi]$	
σ_c, σ_p	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
0.001	135.6	3.0	2005.1	10.8	2246.1	20.1	131.606	1.965	2.586	0.045	14.8	0.4	16.6	0.5
0.003	134.9	2.9	2008.6	16.9	2245.8	19.8	130.148	3.246	2.566	0.079	14.9	0.4	16.7	0.5
0.005	135.2	4.0	2004.8	15.2	2251.5	19.5	130.935	2.561	2.577	0.070	14.8	0.5	16.7	0.6
0.010	135.5	3.8	2000.8	19.0	2244.9	16.2	130.706	2.657	2.571	0.066	14.8	0.5	16.6	0.6
0.025	135.8	2.3	1991.4	17.0	2243.7	23.8	130.294	3.391	2.560	0.064	14.7	0.4	16.5	0.4
0.050	136.5	2.1	1998.6	20.3	2246.6	17.2	130.075	2.602	2.552	0.034	14.6	0.3	16.5	0.3
0.100	137.0	3.3	1996.1	23.9	2255.5	25.5	130.452	2.525	2.560	0.070	14.6	0.5	16.5	0.6
0.250	139.8	3.9	1989.6	19.2	2234.8	18.7	129.376	1.364	2.522	0.046	14.2	0.5	16.0	0.6
0.500	141.1	3.8	1982.6	16.4	2257.7	20.2	131.515	1.954	2.557	0.065	14.1	0.5	16.0	0.6
0.750	141.1	4.1	1977.1	16.8	2278.5	33.7	133.400	3.045	2.598	0.081	14.0	0.5	16.2	0.6
1.000	141.2	5.6	1976.9	27.2	2304.9	29.4	135.434	1.530	2.640	0.067	14.0	0.7	16.3	0.8

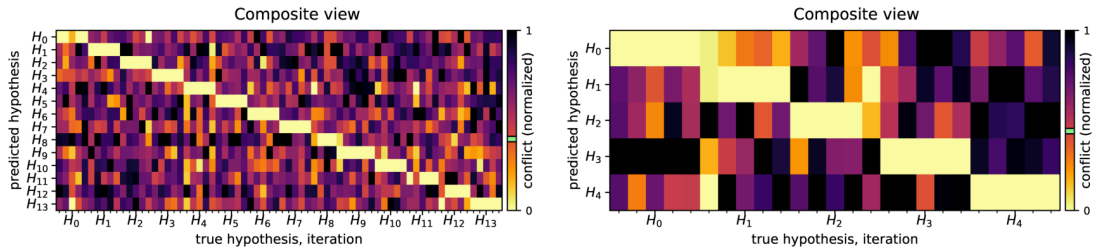
(a) Actuator and sensor noise evaluation on $H_1^{\oplus}SE(2)$.

Noise	$\Lambda[\Xi]$		$\mathcal{S}$		$\mathcal{B}$		T_{CPU}		T_{WALL}		$\mathcal{S}/\Lambda[\Xi]$		$\mathcal{B}/\Lambda[\Xi]$	
σ_c, σ_p	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
0.001	103.6	8.1	2045.3	46.4	2315.6	48.0	143.600	13.765	2.959	0.256	19.9	2.1	22.5	2.3
0.003	104.6	5.1	2048.9	35.1	2297.5	50.6	137.761	7.988	2.846	0.173	19.6	1.2	22.0	1.5
0.005	103.1	7.5	2048.2	59.9	2323.8	68.1	142.129	11.784	2.946	0.262	20.0	1.9	22.7	2.2
0.010	105.9	4.1	2037.0	45.6	2308.0	16.6	137.862	4.519	2.856	0.079	19.3	1.0	21.8	0.9
0.025	106.4	8.0	2033.5	43.1	2296.9	26.2	137.699	5.561	2.850	0.124	19.2	1.5	21.7	1.7
0.050	109.5	6.8	2018.6	42.2	2254.4	33.0	129.666	5.753	2.725	0.163	18.5	1.4	20.6	1.4
0.100	113.9	5.9	2014.0	35.1	2275.8	44.9	134.281	4.445	2.783	0.100	17.7	1.2	20.0	1.1
0.250	119.9	8.2	1998.5	48.6	2252.4	57.1	131.489	6.948	2.690	0.161	16.8	1.5	18.9	1.8
0.500	124.0	10.4	1992.9	37.9	2278.1	89.4	134.166	11.240	2.717	0.285	16.2	1.6	18.5	2.3
0.750	128.0	6.0	1972.8	28.6	2283.1	47.4	131.083	5.657	2.650	0.152	15.4	0.9	17.9	1.0
1.000	126.6	10.2	1965.7	51.4	2315.8	44.6	135.161	3.637	2.747	0.147	15.6	1.7	18.4	1.8

(b) Actuator and sensor noise evaluation on $H_2^{\oplus}SE(2)$.

Table D.32: Actuator and sensor noise evaluation on $H_1^{\oplus}SE(2)$ and $H_2^{\oplus}SE(2)$. Sensorimotor map properties and timing data.

D.8.6 Sensor Permutations With Occupancy on $H_1^{\oplus}SE(2)$ and $H_2^{\oplus}SE(2)$



(a) Evaluation on $H_1^{\oplus}SE(2)$.

(b) Evaluation on $H_1^{\oplus}SE(2)$.

Figure D.50: Sensor permutations with occupancy on $H_1^{\oplus}SE(2)$ and $H_2^{\oplus}SE(2)$. All sensors disabled with $p_L[\circ]$, $p_P[\circ]$, $p_R[\circ]$ and $p_T[\circ]$. Occupancy data enabled with $p_O[\bullet]$.

Sensors				$\Delta[\Xi]/\sum I$		Accuracy
p_L	p_P	p_R	p_T	μ	σ	-
○	○	○	○	0.397	0.273	0.91
○	○	○	●	0.313	0.249	1.00
○	○	●	○	0.376	0.266	1.00
○	○	●	●	0.326	0.253	1.00
○	●	○	○	0.354	0.265	1.00
○	●	○	●	0.308	0.248	1.00
○	●	●	○	0.346	0.257	1.00
○	●	●	●	0.317	0.248	1.00
●	○	○	○	0.355	0.260	1.00
●	○	○	●	0.314	0.250	1.00
●	○	●	○	0.360	0.265	1.00
●	○	●	●	0.327	0.256	1.00
●	●	○	○	0.330	0.255	1.00
●	●	○	●	0.303	0.249	1.00
●	●	●	○	0.337	0.256	1.00
●	●	●	●	0.314	0.251	1.00

(a) Sensor permutations with occupancy on $H_1^{\oplus}SE(2)$.

Sensors				$\Delta[\Xi]/\sum I$		Accuracy
p_L	p_P	p_R	p_T	μ	σ	-
○	○	○	○	0.441	0.375	0.92
○	○	○	●	0.364	0.361	1.00
○	○	●	○	0.400	0.365	1.00
○	○	●	●	0.363	0.361	1.00
○	●	○	○	0.422	0.372	0.96
○	●	○	●	0.369	0.363	1.00
○	●	●	○	0.398	0.365	1.00
○	●	●	●	0.364	0.364	1.00
●	○	○	○	0.394	0.365	1.00
●	○	○	●	0.369	0.358	1.00
●	○	●	○	0.385	0.362	1.00
●	○	●	●	0.368	0.359	1.00
●	●	○	○	0.393	0.364	1.00
●	●	○	●	0.370	0.360	1.00
●	●	●	○	0.386	0.361	1.00
●	●	●	●	0.369	0.359	1.00

(b) Sensor permutations with occupancy on $H_2^{\oplus}SE(2)$.

Table D.33: Sensor permutations with occupancy on $H_1^{\oplus}SE(2)$ and $H_2^{\oplus}SE(2)$. Occupancy data enabled with $p_O[\bullet]$. Intrinsic conflict (μ and σ) and classification accuracy.

Sensor	Accuracy ●		Accuracy ○		Difference ● − ○	
-	μ	σ	μ	σ	μ	σ
p_L	1.000	0.000	0.989	0.030	0.011	0.030
p_P	1.000	0.000	0.989	0.030	0.011	0.030
p_R	1.000	0.000	0.989	0.030	0.011	0.030
p_T	1.000	0.000	0.989	0.030	0.011	0.030

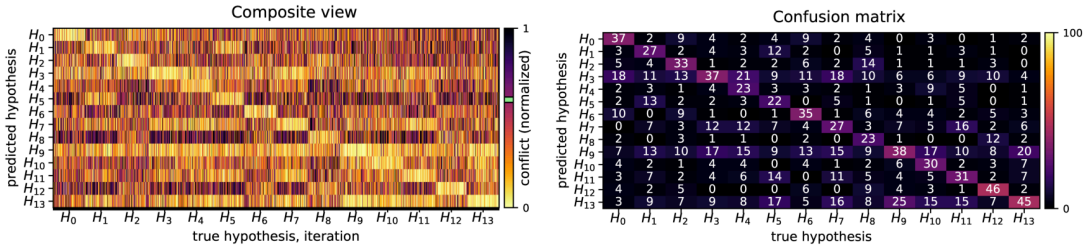
(a) Sensor permutations with occupancy on $H_1^{\oplus}SE(2)$.

Sensor	Accuracy ●		Accuracy ○		Difference ● − ○	
-	μ	σ	μ	σ	μ	σ
p_L	1.000	0.000	0.985	0.028	0.015	0.028
p_P	0.995	0.013	0.990	0.026	0.005	0.030
p_R	1.000	0.000	0.985	0.028	0.015	0.028
p_T	1.000	0.000	0.985	0.028	0.015	0.028

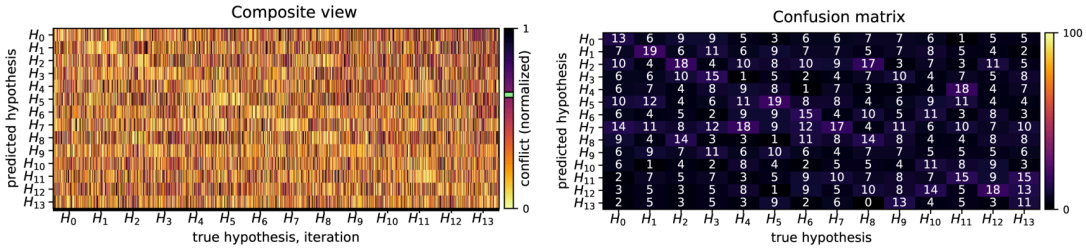
(b) Sensor permutations with occupancy on $H_2^{\oplus}SE(2)$.

Table D.34: Sensor permutations with occupancy on $H_1^{\oplus}SE(2)$ and $H_2^{\oplus}SE(2)$. Classification accuracy evaluated per sensor. Classification impact for each sensor calculated as difference between enabled ● and disabled ○ cases. Occupancy data enabled with $p_O[\bullet]$.

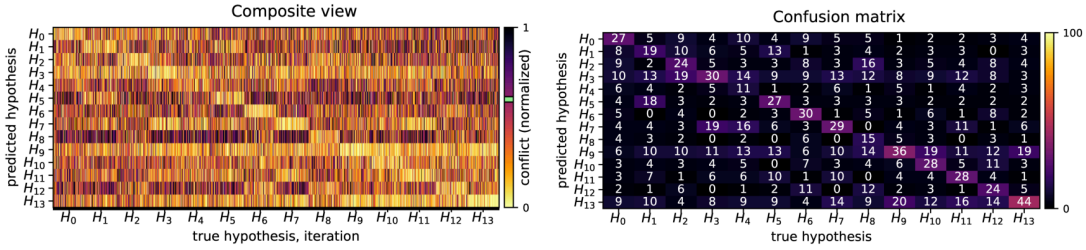
D.8.7 Sensor Permutations Without Occupancy on $H_1^{\oplus}SE(2)$ and $H_2^{\oplus}SE(2)$



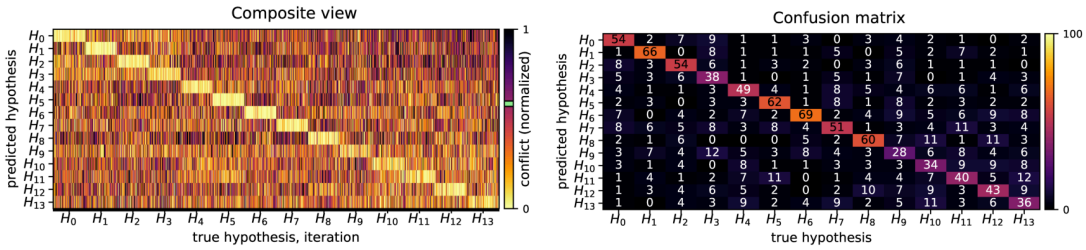
(a) Evaluation with $(p_L[\bullet], p_P[\circ], p_R[\circ], p_T[\circ])$.



(b) Evaluation with $(p_L[\circ], p_P[\bullet], p_R[\circ], p_T[\circ])$.

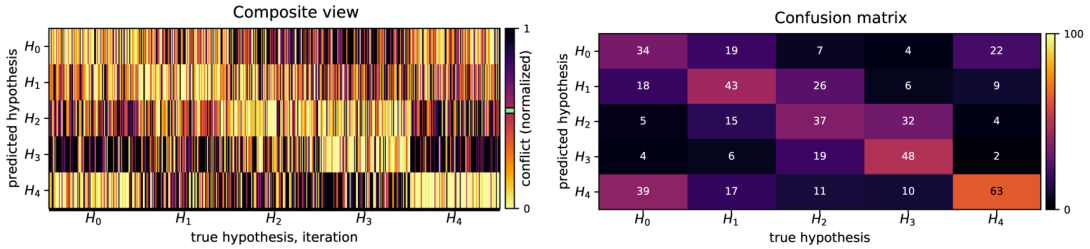


(c) Evaluation with $(p_L[\circ], p_P[\circ], p_R[\bullet], p_T[\circ])$.

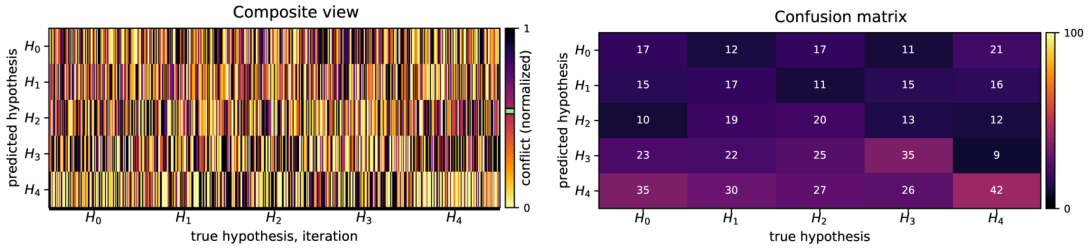


(d) Evaluation with $(p_L[\circ], p_P[\circ], p_R[\circ], p_T[\bullet])$.

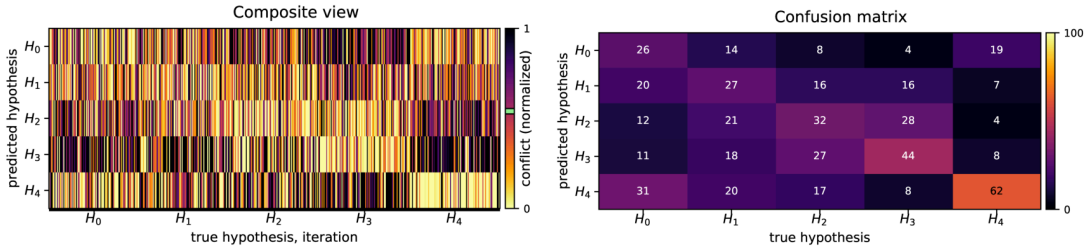
Figure D.51: Sensor permutations without occupancy on $H_1^{\oplus}SE(2)$. Occupancy data disabled with $p_O[\circ]$.



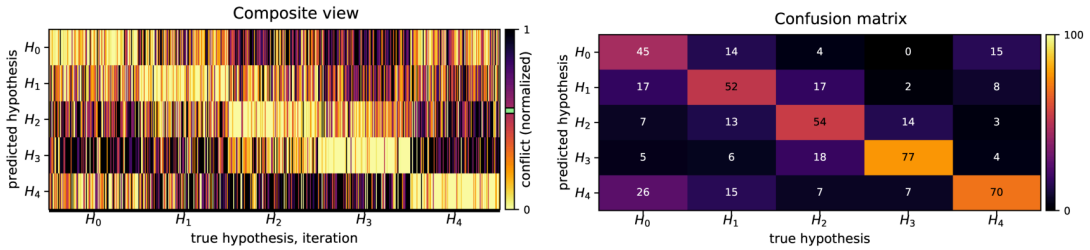
(a) Evaluation with ($p_L[\bullet]$, $p_P[\circ]$, $p_R[\circ]$, $p_T[\circ]$).



(b) Evaluation with ($p_L[\circ]$, $p_P[\bullet]$, $p_R[\circ]$, $p_T[\circ]$).



(c) Evaluation with ($p_L[\circ]$, $p_P[\circ]$, $p_R[\bullet]$, $p_T[\circ]$).



(d) Evaluation with ($p_L[\circ]$, $p_P[\circ]$, $p_R[\circ]$, $p_T[\bullet]$).

Figure D.52: Sensor permutations without occupancy on $\mathrm{H}_2^{\oplus}\mathrm{SE}(2)$. Occupancy data disabled with $p_O[\circ]$.

Sensors				$\Delta[\Xi]/\sum I$		Accuracy
p_L	p_P	p_R	p_T	μ	σ	-
○	○	○	○	nan	nan	0.07
○	○	○	●	0.585	0.299	0.49
○	○	●	○	0.600	0.299	0.27
○	○	●	●	0.574	0.298	0.51
○	●	○	○	0.631	0.310	0.14
○	●	○	●	0.575	0.298	0.45
○	●	●	○	0.593	0.300	0.27
○	●	●	●	0.569	0.298	0.48
●	○	○	○	0.602	0.302	0.32
●	○	○	●	0.571	0.298	0.55
●	○	●	○	0.591	0.301	0.34
●	○	●	●	0.570	0.299	0.54
●	●	○	○	0.588	0.301	0.30
●	●	○	●	0.566	0.298	0.51
●	●	●	○	0.583	0.301	0.33
●	●	●	●	0.565	0.299	0.51

(a) Sensor permutations without occupancy on $\mathrm{H}_1^{\oplus}\mathrm{SE}(2)$.

Sensors				$\Delta[\Xi]/\sum I$		Accuracy
p_L	p_P	p_R	p_T	μ	σ	-
○	○	○	○	nan	nan	0.20
○	○	○	●	0.549	0.384	0.60
○	○	●	○	0.540	0.383	0.38
○	○	●	●	0.548	0.383	0.61
○	●	○	○	0.538	0.390	0.26
○	●	○	●	0.541	0.386	0.56
○	●	●	○	0.534	0.382	0.37
○	●	●	●	0.539	0.383	0.57
●	○	○	○	0.544	0.384	0.45
●	○	○	●	0.544	0.384	0.63
●	○	●	○	0.536	0.384	0.45
●	○	●	●	0.542	0.383	0.61
●	●	○	○	0.539	0.384	0.44
●	●	○	●	0.540	0.384	0.61
●	●	●	○	0.532	0.384	0.46
●	●	●	●	0.537	0.384	0.60

(b) Sensor permutations without occupancy on $\mathrm{H}_2^{\oplus}\mathrm{SE}(2)$.

Table D.35: Sensor permutations without occupancy on $\mathrm{H}_1^{\oplus}\mathrm{SE}(2)$ and $\mathrm{H}_2^{\oplus}\mathrm{SE}(2)$. Occupancy data disabled with $p_O[\circ]$. Intrinsic conflict (μ and σ) and classification accuracy.

Sensor	Accuracy ●		Accuracy ○		Difference ● − ○	
-	μ	σ	μ	σ	μ	σ
p_L	0.425	0.104	0.335	0.160	0.090	0.191
p_P	0.374	0.126	0.386	0.157	-0.013	0.201
p_R	0.406	0.107	0.354	0.166	0.052	0.198
p_T	0.505	0.030	0.255	0.092	0.250	0.096

(a) Sensor permutations without occupancy on $\mathrm{H}_1^{\oplus}\mathrm{SE}(2)$.

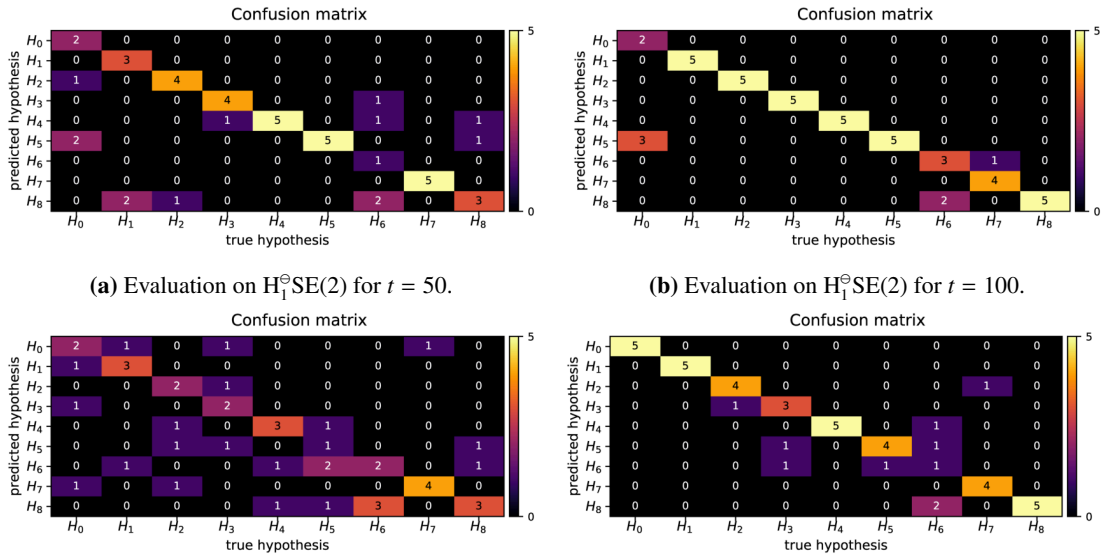
Sensor	Accuracy ●		Accuracy ○		Difference ● − ○	
-	μ	σ	μ	σ	μ	σ
p_L	0.531	0.082	0.444	0.152	0.088	0.172
p_P	0.484	0.116	0.491	0.141	-0.008	0.183
p_R	0.506	0.096	0.469	0.153	0.037	0.181
p_T	0.599	0.021	0.376	0.091	0.223	0.094

(b) Sensor permutations without occupancy on $\mathrm{H}_2^{\oplus}\mathrm{SE}(2)$.

Table D.36: Sensor permutations without occupancy on $\mathrm{H}_1^{\oplus}\mathrm{SE}(2)$ and $\mathrm{H}_2^{\oplus}\mathrm{SE}(2)$. Classification accuracy evaluated per sensor. Classification impact for each sensor calculated as difference between enabled ● and disabled ○ cases. Occupancy data disabled with $p_O[\circ]$.

D.9 Scenario Vb : Nonholonomic Classification on SE(2)

D.9.1 Intermediate Conflict Calculation on $H_1^{\ominus}SE(2)$ and $H_2^{\ominus}SE(2)$



(a) Evaluation on $H_1^{\ominus}SE(2)$ for $t = 50$.

(b) Evaluation on $H_1^{\ominus}SE(2)$ for $t = 100$.

(c) Evaluation on $H_2^{\ominus}SE(2)$ for $t = 50$.

(d) Evaluation on $H_2^{\ominus}SE(2)$ for $t = 100$.

Figure D.53: Intermediate conflict calculation on $H_1^{\ominus}SE(2)$ and $H_2^{\ominus}SE(2)$,for $t = 50$ and $t = 100$.

Timestep	$\Delta[\Xi]/\sum I$		Accuracy
-	μ	σ	-
50	0.573	0.371	0.71
100	0.515	0.355	0.87
150	0.480	0.365	0.93
200	0.456	0.364	0.96
250	0.441	0.360	0.93
300	0.435	0.357	1.00
350	0.428	0.354	1.00
400	0.418	0.355	1.00
450	0.411	0.355	1.00
500	0.397	0.353	1.00
550	0.393	0.352	1.00
600	0.388	0.354	1.00
650	0.384	0.352	1.00
700	0.387	0.349	1.00
750	0.384	0.348	1.00
800	0.382	0.345	1.00
850	0.381	0.343	1.00
900	0.382	0.342	1.00
950	0.380	0.343	1.00
1000	0.376	0.345	1.00

(a) Evaluation on $H_1^{\ominus}SE(2)$.

Timestep	$\Delta[\Xi]/\sum I$		Accuracy
-	μ	σ	-
50	0.570	0.348	0.49
100	0.525	0.333	0.80
150	0.508	0.328	0.91
200	0.474	0.326	0.89
250	0.432	0.330	0.93
300	0.415	0.329	0.93
350	0.414	0.325	0.93
400	0.414	0.321	0.98
450	0.405	0.319	0.96
500	0.386	0.314	1.00
550	0.379	0.312	1.00
600	0.372	0.311	1.00
650	0.363	0.309	1.00
700	0.357	0.309	1.00
750	0.346	0.310	1.00
800	0.340	0.310	1.00
850	0.339	0.307	1.00
900	0.334	0.308	1.00
950	0.333	0.306	1.00
1000	0.331	0.305	1.00

(b) Evaluation on $H_2^{\ominus}SE(2)$.

Table D.37: Intermediate conflict calculation on $H_1^{\ominus}SE(2)$ and $H_2^{\ominus}SE(2)$ for $t = 50$ to $t = 1000$. Intrinsic conflict (μ and σ) and classification accuracy.

Timestep	$\Lambda[\Xi]$		$\mathcal{S}$		$\mathcal{B}$		T_{CPU}		T_{WALL}		$\mathcal{S}/\Lambda[\Xi]$		$\mathcal{B}/\Lambda[\Xi]$	
-	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
50	8.1	0.6	86.8	4.5	88.6	3.6	2.762	0.381	0.058	0.008	10.7	1.4	11.0	1.3
100	14.2	1.3	180.3	5.3	186.4	2.9	7.834	1.448	0.173	0.027	12.8	1.3	13.2	1.3
150	19.2	0.9	274.1	4.0	287.3	3.6	12.856	1.451	0.295	0.030	14.3	0.9	15.0	0.9
200	23.6	1.5	373.5	5.2	393.3	5.7	19.260	1.878	0.444	0.033	15.8	1.1	16.7	1.0
250	28.6	2.3	472.1	8.2	500.7	10.9	25.716	1.435	0.590	0.021	16.6	1.4	17.6	1.2
300	32.5	3.2	569.8	7.7	606.8	12.2	33.186	2.063	0.755	0.023	17.7	1.7	18.8	1.6
350	38.0	3.3	665.3	7.7	712.4	15.7	40.055	2.416	0.894	0.032	17.6	1.5	18.8	1.3
400	42.3	3.7	761.3	7.4	817.4	19.5	46.378	2.449	1.025	0.028	18.1	1.6	19.4	1.3
450	46.1	4.0	859.7	8.5	925.2	22.1	53.708	2.208	1.178	0.029	18.7	1.6	20.1	1.4
500	50.6	4.0	959.8	6.9	1034.0	19.7	61.345	2.509	1.331	0.032	19.1	1.6	20.5	1.3
550	53.9	4.7	1062.5	9.2	1145.4	21.1	69.235	3.078	1.496	0.044	19.8	1.8	21.4	1.5
600	57.6	4.8	1165.5	11.3	1257.0	20.2	77.396	2.858	1.657	0.045	20.3	1.8	21.9	1.6
650	61.8	5.2	1265.9	10.6	1369.1	14.4	86.541	3.375	1.840	0.051	20.6	1.9	22.3	1.7
700	65.2	5.4	1365.9	13.2	1479.5	18.4	94.624	3.836	2.005	0.067	21.1	1.8	22.8	1.6
750	69.2	5.7	1471.0	15.2	1593.6	21.2	104.413	3.585	2.202	0.056	21.4	1.8	23.2	1.7
800	72.5	5.8	1573.7	18.1	1703.8	22.9	114.296	4.424	2.401	0.067	21.8	1.8	23.6	1.7
850	76.7	5.5	1675.6	17.4	1813.8	22.5	122.989	3.031	2.566	0.044	22.0	1.7	23.7	1.5
900	80.7	5.6	1778.8	20.8	1926.7	28.2	132.697	2.097	2.750	0.032	22.1	1.6	24.0	1.5
950	84.5	6.1	1882.1	20.3	2039.2	27.5	142.820	2.330	2.943	0.037	22.4	1.7	24.2	1.6
1000	87.9	6.4	1982.6	21.9	2151.4	26.4	152.065	2.351	3.121	0.041	22.7	1.7	24.6	1.6

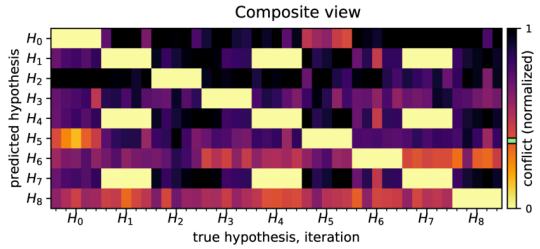
(a) Intermediate conflict calculation on $\mathrm{H}_1^{\ominus}\mathrm{SE}(2)$.

Timestep	$\Lambda[\Xi]$		$\mathcal{S}$		$\mathcal{B}$		T_{CPU}		T_{WALL}		$\mathcal{S}/\Lambda[\Xi]$		$\mathcal{B}/\Lambda[\Xi]$	
-	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
50	8.7	0.9	84.8	4.8	87.4	5.3	2.979	0.550	0.061	0.011	9.9	1.5	10.2	1.5
100	14.9	1.3	176.5	5.4	185.4	3.9	7.810	0.927	0.170	0.021	11.9	1.3	12.5	1.2
150	20.1	1.8	269.2	4.4	289.7	4.7	13.001	1.638	0.292	0.030	13.5	1.3	14.5	1.3
200	24.4	2.1	367.3	6.5	397.4	6.2	18.730	1.814	0.430	0.032	15.1	1.5	16.4	1.4
250	29.2	2.6	464.8	10.3	505.2	10.2	25.203	1.413	0.580	0.016	16.0	1.7	17.4	1.6
300	32.9	3.3	563.0	12.3	614.4	11.3	31.780	1.401	0.731	0.023	17.3	1.9	18.8	1.9
350	38.2	2.7	660.2	10.0	725.1	14.4	39.492	1.195	0.892	0.025	17.4	1.3	19.1	1.2
400	42.4	2.9	757.2	12.3	835.4	19.3	46.352	1.293	1.042	0.023	17.9	1.3	19.8	1.3
450	46.0	3.6	856.4	14.9	950.1	21.4	54.405	1.801	1.213	0.032	18.7	1.6	20.7	1.7
500	49.8	3.4	958.9	13.4	1068.4	20.0	62.538	1.681	1.387	0.038	19.3	1.4	21.5	1.4
550	52.6	3.6	1062.8	13.5	1186.2	24.2	71.066	2.764	1.577	0.043	20.3	1.5	22.6	1.4
600	55.7	3.6	1167.2	13.3	1304.4	27.2	80.450	3.873	1.761	0.058	21.0	1.5	23.5	1.4
650	59.8	4.1	1269.2	15.5	1420.1	29.4	89.571	5.353	1.928	0.102	21.3	1.6	23.8	1.6
700	62.7	4.0	1371.8	20.2	1536.5	37.4	98.859	6.885	2.105	0.125	22.0	1.7	24.6	1.6
750	66.2	4.7	1478.2	21.4	1654.9	40.8	109.217	7.774	2.308	0.138	22.4	1.9	25.1	1.9
800	69.4	5.0	1583.3	23.2	1769.3	40.2	117.957	8.808	2.484	0.161	22.9	2.0	25.6	2.0
850	73.5	5.1	1684.6	24.2	1880.0	39.7	126.474	8.317	2.649	0.142	23.0	1.9	25.7	2.0
900	77.7	4.9	1786.5	26.5	1992.8	43.5	136.386	9.612	2.848	0.165	23.1	1.8	25.7	1.8
950	81.8	4.9	1887.8	26.1	2106.3	41.8	146.017	9.801	3.035	0.163	23.2	1.7	25.8	1.7
1000	84.8	5.2	1987.2	28.4	2219.6	44.1	154.486	9.490	3.209	0.154	23.5	1.8	26.2	1.8

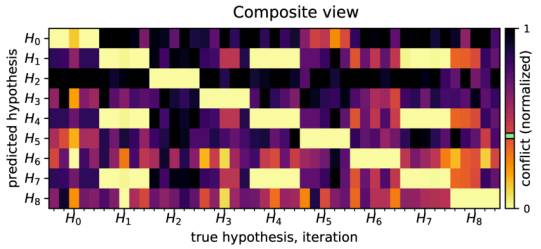
(b) Intermediate conflict calculation on $\mathrm{H}_2^{\ominus}\mathrm{SE}(2)$.

Table D.38: Intermediate conflict calculation on $\mathrm{H}_1^{\ominus}\mathrm{SE}(2)$ and $\mathrm{H}_2^{\ominus}\mathrm{SE}(2)$ for $t = 50$ to $t = 1000$. Sensorimotor map properties and timing data.

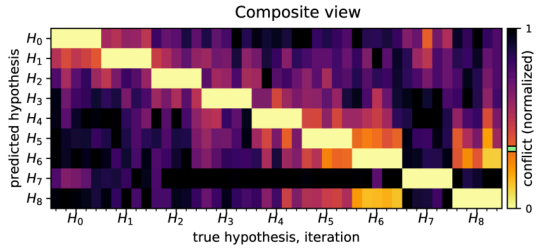
D.9.2 Subset Conflict Calculation on Hypothesis Set $\mathrm{H}_1^{\ominus}\mathrm{SE}(2)$ and $\mathrm{H}_2^{\ominus}\mathrm{SE}(2)$



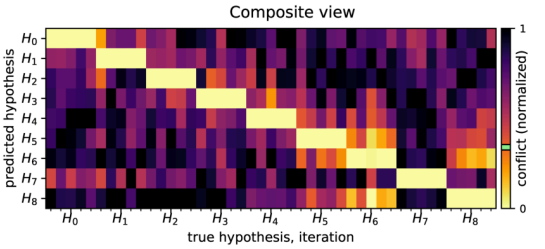
(a) Evaluation on $\mathrm{H}_1^{\ominus}\mathrm{SE}(2)$ for $n = 1$ and $r = 1.0$.



(b) Evaluation on $\mathrm{H}_1^{\ominus}\mathrm{SE}(2)$ for $n = 10$ and $r = 0.1$.



(c) Evaluation on $\mathrm{H}_2^{\ominus}\mathrm{SE}(2)$ for $n = 1$ and $r = 1.0$.



(d) Evaluation on $\mathrm{H}_2^{\ominus}\mathrm{SE}(2)$ for $n = 10$ and $r = 0.1$.

Figure D.54: Subset conflict calculation on $\mathrm{H}_1^{\ominus}\mathrm{SE}(2)$ and $\mathrm{H}_2^{\ominus}\mathrm{SE}(2)$, for $(n = 1, r = 1.0)$ and $(n = 10, r = 0.1)$.

Subsets	Ratio	$\Delta[\Xi]/\sum I$		Accuracy
n	r	μ	σ	-
1	1.000	0.376	0.345	1.00
2	0.500	0.329	0.348	1.00
3	0.333	0.310	0.351	1.00
4	0.250	0.314	0.349	1.00
5	0.200	0.318	0.348	1.00
6	0.166	0.333	0.349	1.00
7	0.142	0.344	0.349	1.00
8	0.125	0.356	0.352	1.00
9	0.111	0.386	0.349	1.00
10	0.100	0.403	0.348	0.96

(a) Subset conflict calculation for $\mathrm{H}_1^{\ominus}\mathrm{SE}(2)$. Intrinsic conflict ($\mu$ and σ) and classification accuracy.

Subsets	Ratio	T_{CPU}		T_{WALL}	
n	r	μ	σ	μ	σ
1	1.000	148.437	2.902	3.038	0.062
2	0.500	116.370	3.495	2.286	0.051
3	0.333	116.738	4.105	2.049	0.032
4	0.250	124.364	7.576	2.023	0.092
5	0.200	133.042	7.642	2.053	0.095
6	0.166	137.938	7.081	2.059	0.090
7	0.142	147.173	8.850	2.136	0.109
8	0.125	157.324	7.700	2.249	0.114
9	0.111	155.798	12.401	2.199	0.186
10	0.100	172.959	15.167	2.421	0.218

(b) Subset conflict calculation for $\mathrm{H}_1^{\ominus}\mathrm{SE}(2)$. Timing data.

Subsets	Ratio	$\Delta[\Xi]/\sum I$		Accuracy
n	r	μ	σ	-
1	1.000	0.331	0.305	1.00
2	0.500	0.310	0.306	1.00
3	0.333	0.298	0.310	1.00
4	0.250	0.301	0.309	1.00
5	0.200	0.305	0.308	1.00
6	0.166	0.309	0.307	1.00
7	0.142	0.314	0.307	1.00
8	0.125	0.316	0.308	1.00
9	0.111	0.330	0.308	0.98
10	0.100	0.340	0.308	0.98

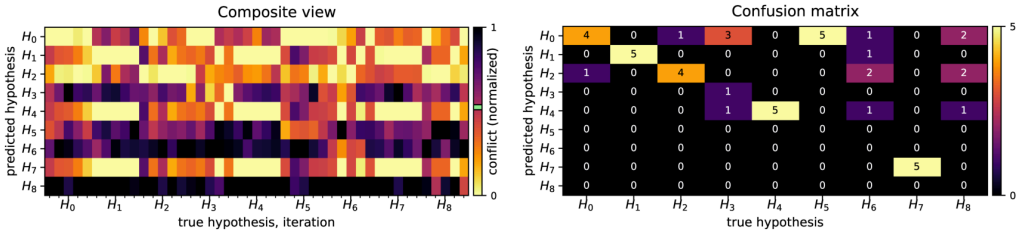
(c) Subset conflict calculation for $\mathrm{H}_2^{\ominus}\mathrm{SE}(2)$. Intrinsic conflict ($\mu$ and σ) and classification accuracy.

Subsets	Ratio	T_{CPU}		T_{WALL}	
n	r	μ	σ	μ	σ
1	1.000	150.489	9.778	3.108	0.162
2	0.500	117.589	5.579	2.295	0.121
3	0.333	117.107	5.588	2.026	0.095
4	0.250	124.260	6.884	1.990	0.097
5	0.200	128.661	4.480	1.942	0.069
6	0.166	133.308	5.258	1.931	0.071
7	0.142	141.133	7.713	1.993	0.118
8	0.125	146.890	5.970	2.034	0.096
9	0.111	144.590	4.930	1.976	0.067
10	0.100	155.238	2.237	2.101	0.028

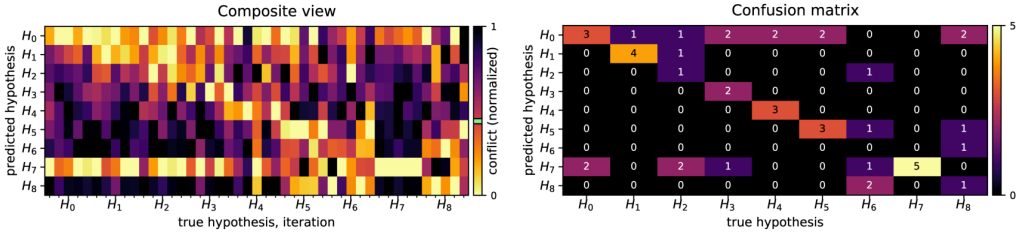
(d) Subset conflict calculation for $\mathrm{H}_2^{\ominus}\mathrm{SE}(2)$. Timing data.

Table D.39: Subset conflict calculation on $\mathrm{H}_1^{\ominus}\mathrm{SE}(2)$ and $\mathrm{H}_2^{\ominus}\mathrm{SE}(2)$ for $(n = 1, r = 1.0)$ to $(n = 10, r = 0.1)$. Intrinsic conflict (μ and σ) and classification accuracy.Timing data.

D.9.3 Sensor Noise Evaluation on $H_1^{\ominus}SE(2)$ and $H_2^{\ominus}SE(2)$



(a) Noisy sensors on $H_1^{\ominus}SE(2)$ with σ_p set to 1.000.



(b) Noisy sensors on $H_1^{\ominus}SE(2)$ with σ_p set to 1.000.

Figure D.55: Sensor noise evaluation on $H_1^{\ominus}SE(2)$ and $H_2^{\ominus}SE(2)$ for $\sigma_p = 1.000$.

Noise	$\Delta[\Xi]/\sum I$		Accuracy
σ_p	μ	σ	-
0.001	0.376	0.345	1.00
0.003	0.376	0.345	1.00
0.005	0.376	0.345	1.00
0.010	0.376	0.345	1.00
0.025	0.375	0.345	1.00
0.050	0.371	0.345	1.00
0.100	0.358	0.347	1.00
0.250	0.318	0.351	1.00
0.500	0.437	0.351	0.80
0.750	0.506	0.354	0.60
1.000	0.525	0.353	0.53

(a) Sensor noise evaluation for $H_1^{\ominus}SE(2)$. Intrinsic conflict (μ and σ) and classification accuracy.

Noise	T_{CPU}		T_{WALL}	
σ_p	μ	σ	μ	σ
0.001	147.572	4.225	3.003	0.083
0.003	147.409	3.605	2.997	0.075
0.005	147.177	2.410	2.996	0.035
0.010	146.636	2.529	2.977	0.046
0.025	146.605	2.706	2.983	0.057
0.050	147.282	3.032	2.996	0.053
0.100	146.481	3.385	2.974	0.070
0.250	146.606	3.914	2.980	0.064
0.500	147.198	3.765	2.991	0.071
0.750	146.856	3.080	2.993	0.059
1.000	146.620	3.022	2.979	0.056

(b) Sensor noise evaluation for $H_1^{\ominus}SE(2)$. Timing data.

Noise	$\Delta[\Xi]/\sum I$		Accuracy
σ_p	μ	σ	-
0.001	0.331	0.305	1.00
0.003	0.331	0.305	1.00
0.005	0.331	0.305	1.00
0.010	0.331	0.305	1.00
0.025	0.331	0.305	1.00
0.050	0.330	0.305	1.00
0.100	0.326	0.305	1.00
0.250	0.321	0.305	1.00
0.500	0.369	0.324	0.93
0.750	0.432	0.342	0.58
1.000	0.438	0.339	0.49

(c) Sensor noise evaluation for $H_2^{\ominus}SE(2)$. Intrinsic conflict (μ and σ) and classification accuracy.

Noise	T_{CPU}		T_{WALL}	
σ_p	μ	σ	μ	σ
0.001	150.766	10.017	3.114	0.170
0.003	149.696	9.667	3.096	0.151
0.005	149.683	8.784	3.093	0.146
0.010	149.487	10.494	3.085	0.174
0.025	149.380	10.082	3.085	0.170
0.050	149.213	8.765	3.082	0.139
0.100	150.696	10.260	3.112	0.164
0.250	149.518	9.733	3.094	0.156
0.500	150.304	9.207	3.106	0.154
0.750	149.729	9.891	3.094	0.166
1.000	149.575	9.617	3.092	0.153

(d) Sensor noise evaluation for $H_2^{\ominus}SE(2)$. Timing data.

Table D.40: Sensor noise evaluation on $H_1^{\ominus}SE(2)$ and $H_2^{\ominus}SE(2)$. Intrinsic conflict (μ and σ) and classification accuracy. Timing data.

D.9.4 Actuator Noise Evaluation on $H_1^{\ominus}SE(2)$ and $H_2^{\ominus}SE(2)$

Noise	$\Delta[\Xi]/\sum I$		Accuracy
σ_c	μ	σ	-
0.001	0.375	0.351	1.00
0.003	0.373	0.349	1.00
0.005	0.380	0.347	1.00
0.010	0.389	0.350	1.00
0.025	0.404	0.349	1.00
0.050	0.425	0.357	1.00
0.100	0.441	0.365	0.82
0.250	0.483	0.356	0.36
0.500	0.426	0.364	0.18
0.750	0.455	0.354	0.13
1.000	0.426	0.349	0.16

(a) Actuator noise evaluation on $H_1^{\ominus}SE(2)$.

Noise	$\Delta[\Xi]/\sum I$		Accuracy
σ_c	μ	σ	-
0.001	0.324	0.305	1.00
0.003	0.335	0.307	1.00
0.005	0.347	0.309	1.00
0.010	0.355	0.310	1.00
0.025	0.379	0.317	0.93
0.050	0.392	0.331	0.91
0.100	0.460	0.339	0.80
0.250	0.527	0.335	0.58
0.500	0.553	0.323	0.44
0.750	0.552	0.327	0.24
1.000	0.513	0.323	0.33

(b) Actuator noise evaluation on $H_2^{\ominus}SE(2)$.

Table D.41: Actuator noise evaluation on $H_1^{\ominus}SE(2)$ and $H_2^{\ominus}SE(2)$. Intrinsic conflict (μ and σ) and classification accuracy.

Noise	$\Lambda[\Xi]$		$\mathcal{S}$		$\mathcal{B}$		T_{CPU}		T_{WALL}		$\mathcal{S}/\Lambda[\Xi]$		$\mathcal{B}/\Lambda[\Xi]$	
σ_c	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
0.001	86.7	6.3	1979.0	10.9	2161.6	10.4	151.622	8.558	3.074	0.109	22.9	1.7	25.0	1.7
0.003	86.8	5.3	1982.2	13.9	2168.2	27.3	150.994	5.777	3.064	0.092	22.9	1.5	25.0	1.4
0.005	87.9	6.4	1985.4	20.3	2165.2	31.8	151.065	3.636	3.057	0.088	22.7	1.8	24.7	1.8
0.010	88.3	5.6	1992.6	14.6	2180.6	27.1	151.604	4.378	3.077	0.053	22.6	1.5	24.8	1.5
0.025	88.2	6.2	1982.5	14.9	2178.0	20.9	150.855	8.155	3.059	0.127	22.6	1.7	24.8	1.8
0.050	88.3	5.3	1982.5	26.8	2183.4	22.2	152.498	9.145	3.094	0.150	22.5	1.5	24.8	1.5
0.100	89.7	6.0	1978.6	6.4	2175.4	6.7	148.866	5.291	3.027	0.078	22.1	1.5	24.3	1.5
0.250	91.5	5.8	1989.6	12.9	2184.4	30.2	148.931	4.088	3.037	0.053	21.8	1.4	23.9	1.2
0.500	91.5	8.3	1992.0	23.3	2225.2	29.4	158.489	9.990	3.207	0.181	21.9	2.1	24.5	2.4
0.750	94.0	7.4	1988.2	22.9	2241.1	12.3	157.890	8.546	3.205	0.165	21.3	1.9	24.0	2.0
1.000	93.7	9.0	1986.3	13.7	2280.2	22.1	163.381	9.689	3.314	0.147	21.4	2.1	24.5	2.4

(a) Actuator noise evaluation on $H_1^{\ominus}SE(2)$.

Noise	$\Lambda[\Xi]$		$\mathcal{S}$		$\mathcal{B}$		T_{CPU}		T_{WALL}		$\mathcal{S}/\Lambda[\Xi]$		$\mathcal{B}/\Lambda[\Xi]$	
σ_c	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
0.001	87.0	4.7	1976.2	17.1	2221.3	32.9	152.000	7.192	3.124	0.105	22.8	1.4	25.6	1.5
0.003	85.6	2.3	1983.7	16.6	2222.8	59.1	150.436	16.170	3.097	0.260	23.2	0.8	26.0	0.9
0.005	86.1	6.7	1978.6	26.0	2216.9	42.0	149.403	11.618	3.087	0.187	23.1	2.0	25.9	2.0
0.010	85.3	3.6	1979.9	20.4	2229.9	41.0	152.351	7.231	3.130	0.114	23.3	1.2	26.2	1.1
0.025	88.9	4.6	1972.7	27.5	2205.9	41.9	142.434	8.887	2.969	0.154	22.3	1.4	24.9	1.4
0.050	86.2	4.8	1979.6	28.6	2223.0	29.6	146.158	5.836	3.040	0.101	23.0	1.5	25.8	1.3
0.100	88.2	5.2	1973.0	23.1	2221.3	11.3	150.421	5.082	3.105	0.062	22.4	1.5	25.3	1.4
0.250	91.7	3.9	1968.9	16.6	2232.1	23.2	147.618	5.713	3.079	0.085	21.5	1.0	24.4	0.9
0.500	92.3	6.6	1971.3	24.9	2277.6	40.9	152.209	6.256	3.178	0.111	21.5	1.8	24.8	2.0
0.750	95.3	5.5	1960.4	14.2	2270.9	18.1	149.610	5.578	3.147	0.095	20.6	1.3	23.9	1.4
1.000	94.8	6.1	1957.6	12.7	2291.8	32.4	152.945	11.160	3.215	0.184	20.7	1.5	24.3	1.8

(b) Actuator noise evaluation on $H_2^{\ominus}SE(2)$.

Table D.42: Actuator noise evaluation on $H_1^{\ominus}SE(2)$ and $H_2^{\ominus}SE(2)$. Sensorimotor map properties and timing data.

D.9.5 Actuator and Sensor Noise Evaluation on $H_1^{\ominus}SE(2)$ and $H_2^{\ominus}SE(2)$

Noise	$\Delta[\Xi]/\sum I$		Accuracy
σ_c, σ_p	μ	σ	-
0.001	0.375	0.351	1.00
0.003	0.373	0.349	1.00
0.005	0.380	0.347	1.00
0.010	0.389	0.350	1.00
0.025	0.403	0.349	1.00
0.050	0.421	0.357	1.00
0.100	0.429	0.365	0.91
0.250	0.456	0.360	0.73
0.500	0.514	0.349	0.47
0.750	0.561	0.355	0.31
1.000	0.537	0.355	0.31

(a) Actuator and sensor noise evaluation on $H_1^{\ominus}SE(2)$.

Noise	$\Delta[\Xi]/\sum I$		Accuracy
σ_c, σ_p	μ	σ	-
0.001	0.324	0.305	1.00
0.003	0.335	0.307	1.00
0.005	0.347	0.309	1.00
0.010	0.355	0.310	1.00
0.025	0.378	0.317	0.93
0.050	0.393	0.332	0.91
0.100	0.455	0.339	0.84
0.250	0.479	0.339	0.60
0.500	0.472	0.328	0.22
0.750	0.441	0.329	0.29
1.000	0.438	0.321	0.29

(b) Actuator and sensor noise evaluation on $H_2^{\ominus}SE(2)$.

Table D.43: Actuator and sensor noise evaluation on $H_1^{\ominus}SE(2)$ and $H_2^{\ominus}SE(2)$. Intrinsic conflict (μ and σ) and classification accuracy.

Noise	$\Lambda[\Xi]$		$\mathcal{S}$		$\mathcal{B}$		T_{CPU}		T_{WALL}		$\mathcal{S}/\Lambda[\Xi]$		$\mathcal{B}/\Lambda[\Xi]$	
σ_c, σ_p	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
0.001	86.7	6.3	1979.0	10.9	2161.6	10.4	151.747	9.114	3.074	0.125	22.9	1.7	25.0	1.7
0.003	86.8	5.3	1982.2	13.9	2168.2	27.3	150.736	5.942	3.062	0.096	22.9	1.5	25.0	1.4
0.005	87.9	6.4	1985.4	20.3	2165.2	31.8	150.438	3.344	3.043	0.087	22.7	1.8	24.7	1.8
0.010	88.3	5.6	1992.6	14.6	2180.6	27.1	151.316	5.171	3.069	0.074	22.6	1.5	24.8	1.5
0.025	88.2	6.2	1982.5	14.9	2178.0	20.9	150.856	8.298	3.062	0.138	22.6	1.7	24.8	1.8
0.050	88.3	5.3	1982.5	26.8	2183.4	22.2	152.444	8.836	3.092	0.142	22.5	1.5	24.8	1.5
0.100	89.7	6.0	1978.6	6.4	2175.4	6.7	149.421	5.054	3.037	0.072	22.1	1.5	24.3	1.5
0.250	91.5	5.8	1989.6	12.9	2184.4	30.2	148.959	4.153	3.041	0.057	21.8	1.4	23.9	1.2
0.500	91.5	8.3	1992.0	23.3	2225.2	29.4	158.314	10.133	3.201	0.182	21.9	2.1	24.5	2.4
0.750	94.0	7.4	1988.2	22.9	2241.1	12.3	157.734	8.268	3.203	0.156	21.3	1.9	24.0	2.0
1.000	93.7	9.0	1986.3	13.7	2280.2	22.1	163.219	9.129	3.306	0.142	21.4	2.1	24.5	2.4

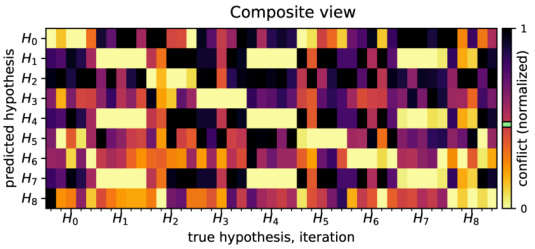
(a) Actuator and sensor noise evaluation on $H_1^{\ominus}SE(2)$.

Noise	$\Lambda[\Xi]$		$\mathcal{S}$		$\mathcal{B}$		T_{CPU}		T_{WALL}		$\mathcal{S}/\Lambda[\Xi]$		$\mathcal{B}/\Lambda[\Xi]$	
σ_c, σ_p	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
0.001	87.0	4.7	1976.2	17.1	2221.3	32.9	149.982	8.824	3.085	0.141	22.8	1.4	25.6	1.5
0.003	85.6	2.3	1983.7	16.6	2222.8	59.1	150.266	15.236	3.090	0.237	23.2	0.8	26.0	0.9
0.005	86.1	6.7	1978.6	26.0	2216.9	42.0	149.065	11.599	3.077	0.179	23.1	2.0	25.9	2.0
0.010	85.3	3.6	1979.9	20.4	2229.9	41.0	150.942	8.015	3.098	0.126	23.3	1.2	26.2	1.1
0.025	88.9	4.6	1972.7	27.5	2205.9	41.9	143.027	8.818	2.980	0.161	22.3	1.4	24.9	1.4
0.050	86.2	4.8	1979.6	28.6	2223.0	29.6	145.851	5.810	3.034	0.093	23.0	1.5	25.8	1.3
0.100	88.2	5.2	1973.0	23.1	2221.3	11.3	150.050	4.805	3.101	0.060	22.4	1.5	25.3	1.4
0.250	91.7	3.9	1968.9	16.6	2232.1	23.2	147.869	5.840	3.083	0.091	21.5	1.0	24.4	0.9
0.500	92.3	6.6	1971.3	24.9	2277.6	40.9	152.086	6.374	3.173	0.117	21.5	1.8	24.8	2.0
0.750	95.3	5.5	1960.4	14.2	2270.9	18.1	149.055	5.389	3.130	0.089	20.6	1.3	23.9	1.4
1.000	94.8	6.1	1957.6	12.7	2291.8	32.4	152.103	11.259	3.192	0.179	20.7	1.5	24.3	1.8

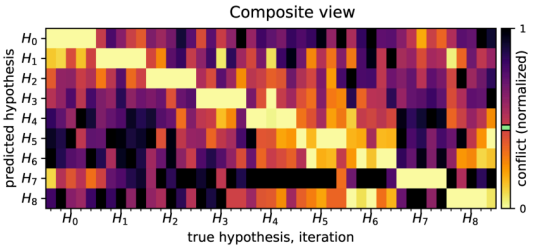
(b) Actuator and sensor noise evaluation on $H_2^{\ominus}SE(2)$.

Table D.44: Actuator and sensor noise evaluation on $H_1^{\ominus}SE(2)$ and $H_2^{\ominus}SE(2)$. Sensorimotor map properties and timing data.

D.9.6 Sensor Permutations With Occupancy on $\mathrm{H}_1^{\ominus}\mathrm{SE}(2)$ and $\mathrm{H}_2^{\ominus}\mathrm{SE}(2)$



(a) Evaluation on $\mathrm{H}_1^{\ominus}\mathrm{SE}(2)$.



(b) Evaluation on $\mathrm{H}_1^{\ominus}\mathrm{SE}(2)$.

Figure D.56: Sensor permutations with occupancy on $\mathrm{H}_1^{\ominus}\mathrm{SE}(2)$ and $\mathrm{H}_2^{\ominus}\mathrm{SE}(2)$. All sensors disabled with $p_L[\circ]$, $p_P[\circ]$, $p_R[\circ]$ and $p_T[\circ]$. Occupancy data enabled with $p_O[\bullet]$.

Sensors				$\Delta[\Xi]/\sum I$		Accuracy
p_L	p_P	p_R	p_T	μ	σ	-
○	○	○	○	0.466	0.362	0.80
○	○	○	●	0.402	0.356	1.00
○	○	●	○	0.468	0.348	0.93
○	○	●	●	0.412	0.349	1.00
○	●	○	○	0.428	0.361	0.96
○	●	○	●	0.399	0.360	1.00
○	●	●	○	0.425	0.354	0.98
○	●	●	●	0.400	0.355	1.00
●	○	○	○	0.431	0.349	1.00
●	○	○	●	0.388	0.354	1.00
●	○	●	○	0.458	0.340	0.98
●	○	●	●	0.409	0.348	1.00
●	●	○	○	0.412	0.351	1.00
●	●	○	●	0.387	0.356	1.00
●	●	●	○	0.424	0.348	1.00
●	●	●	●	0.397	0.353	1.00

(a) Sensor permutations with occupancy on $\mathrm{H}_1^{\ominus}\mathrm{SE}(2)$.

Sensors				$\Delta[\Xi]/\sum I$		Accuracy
p_L	p_P	p_R	p_T	μ	σ	-
○	○	○	○	0.449	0.314	0.89
○	○	○	●	0.398	0.309	0.98
○	○	●	○	0.410	0.319	0.96
○	○	●	●	0.388	0.312	1.00
○	●	○	○	0.436	0.318	0.87
○	●	○	●	0.408	0.312	0.98
○	●	●	○	0.415	0.317	0.98
○	●	●	●	0.399	0.312	1.00
●	○	○	○	0.405	0.317	0.98
●	○	○	●	0.382	0.313	0.98
●	○	●	○	0.392	0.321	0.98
●	○	●	●	0.377	0.315	1.00
●	●	○	○	0.406	0.320	0.96
●	●	○	●	0.391	0.314	0.98
●	●	●	○	0.396	0.318	0.98
●	●	●	●	0.386	0.314	1.00

(b) Sensor permutations with occupancy on $\mathrm{H}_2^{\ominus}\mathrm{SE}(2)$.

Table D.45: Sensor permutations with occupancy on $\mathrm{H}_1^{\ominus}\mathrm{SE}(2)$ and $\mathrm{H}_2^{\ominus}\mathrm{SE}(2)$. Occupancy data enabled with $p_O[\bullet]$. Intrinsic conflict (μ and σ) and classification accuracy.

Sensor	Accuracy •		Accuracy ○		Difference • − ○	
-	μ	σ	μ	σ	μ	σ
p_L	0.998	0.007	0.959	0.065	0.039	0.065
p_P	0.992	0.014	0.964	0.066	0.029	0.067
p_R	0.986	0.023	0.970	0.066	0.016	0.069
p_T	1.000	0.000	0.956	0.063	0.044	0.063

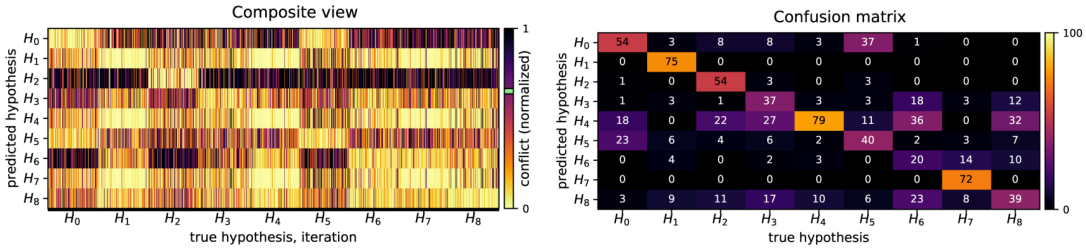
(a) Sensor permutations with occupancy on $\mathrm{H}_1^{\ominus}\mathrm{SE}(2)$.

Sensor	Accuracy •		Accuracy ○		Difference • − ○	
-	μ	σ	μ	σ	μ	σ
p_L	0.982	0.012	0.958	0.047	0.025	0.048
p_P	0.969	0.039	0.971	0.033	-0.002	0.051
p_R	0.988	0.014	0.953	0.043	0.035	0.045
p_T	0.990	0.010	0.950	0.042	0.040	0.043

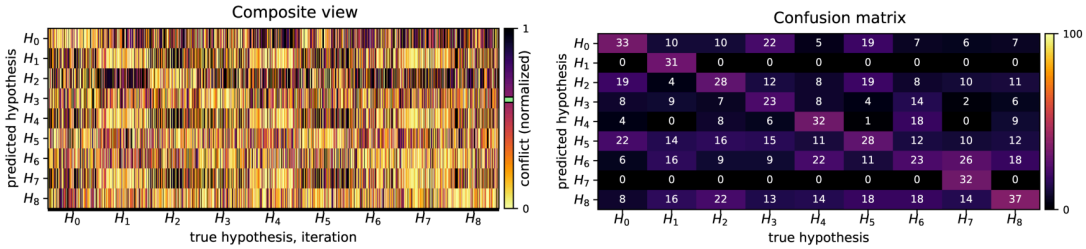
(b) Sensor permutations with occupancy on $\mathrm{H}_2^{\ominus}\mathrm{SE}(2)$.

Table D.46: Sensor permutations with occupancy on $\mathrm{H}_1^{\ominus}\mathrm{SE}(2)$ and $\mathrm{H}_2^{\ominus}\mathrm{SE}(2)$. Classification accuracy evaluated per sensor. Classification impact for each sensor calculated as difference between enabled • and disabled ○ cases. Occupancy data enabled with $p_O[\bullet]$.

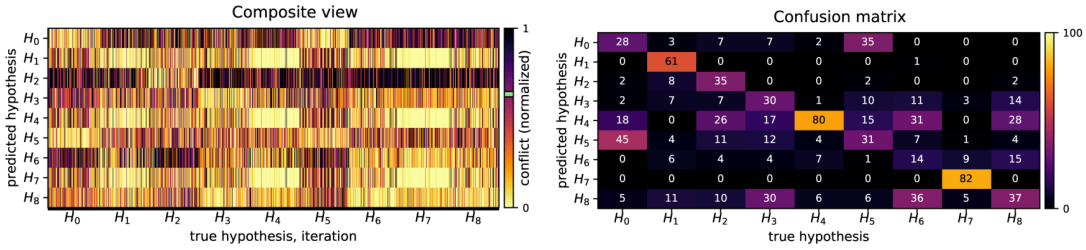
D.9.7 Sensor Permutations Without Occupancy on $\mathrm{H}_1^{\ominus}\mathrm{SE}(2)$ and $\mathrm{H}_2^{\ominus}\mathrm{SE}(2)$



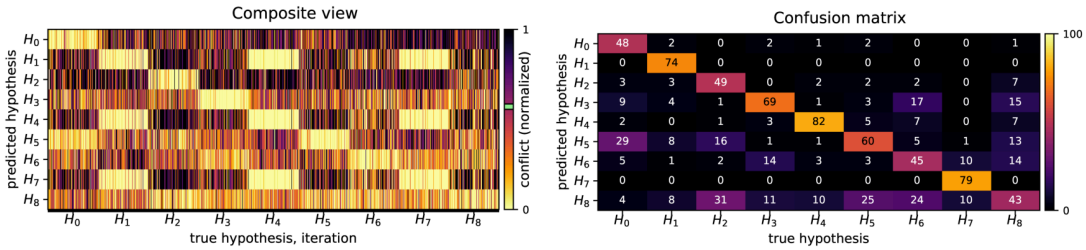
(a) Evaluation with ($p_L[\bullet]$, $p_P[\circ]$, $p_R[\circ]$, $p_T[\circ]$).



(b) Evaluation with ($p_L[\circ]$, $p_P[\bullet]$, $p_R[\circ]$, $p_T[\circ]$).

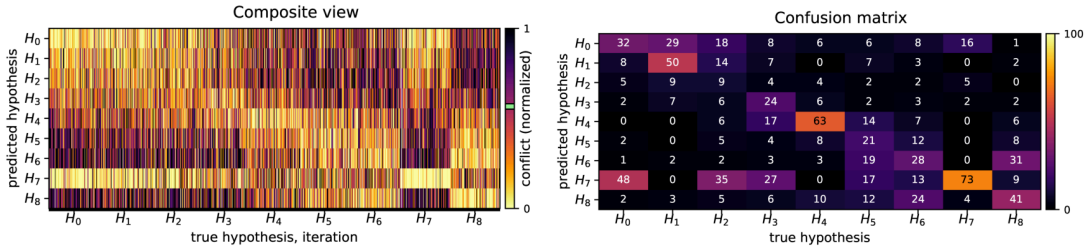


(c) Evaluation with ($p_L[\circ]$, $p_P[\circ]$, $p_R[\bullet]$, $p_T[\circ]$).

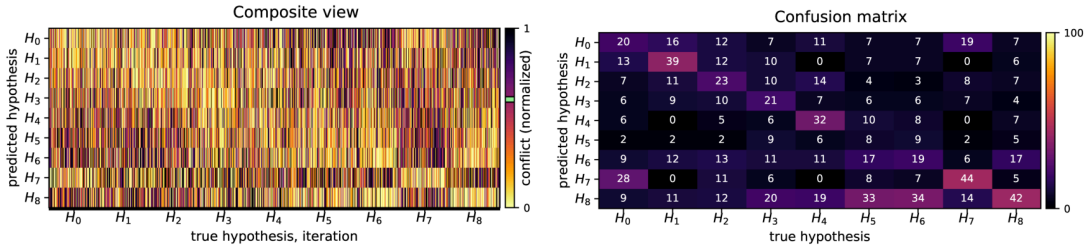


(d) Evaluation with ($p_L[\circ]$, $p_P[\circ]$, $p_R[\circ]$, $p_T[\bullet]$).

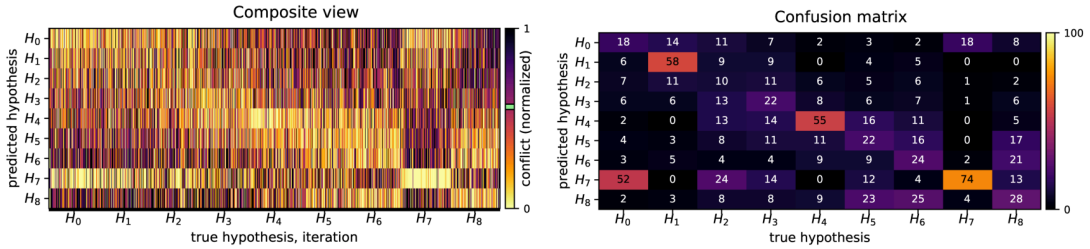
Figure D.57: Sensor permutations without occupancy on $\mathrm{H}_1^{\ominus}\mathrm{SE}(2)$. Occupancy data disabled with $p_O[\circ]$.



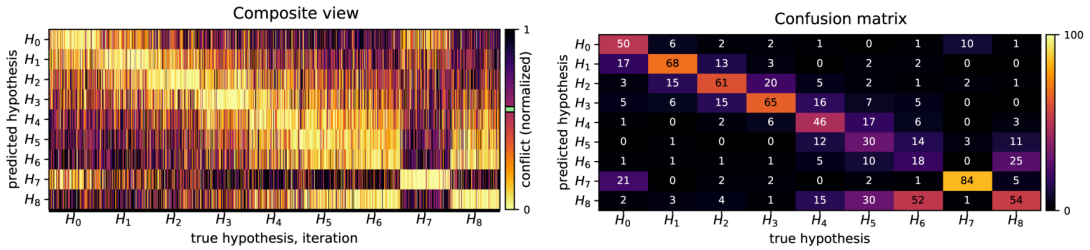
(a) Evaluation with $(p_L[\bullet], p_P[\circ], p_R[\circ], p_T[\circ])$.



(b) Evaluation with $(p_L[\circ], p_P[\bullet], p_R[\circ], p_T[\circ])$.



(c) Evaluation with $(p_L[\circ], p_P[\circ], p_R[\bullet], p_T[\circ])$.



(d) Evaluation with $(p_L[\circ], p_P[\circ], p_R[\circ], p_T[\bullet])$.

Figure D.58: Sensor permutations without occupancy on $\mathrm{H}_2^{\ominus}\mathrm{SE}(2)$. Occupancy data disabled with $p_O[\circ]$.

Sensors				$\Delta[\Xi]/\sum I$		Accuracy
p_L	p_P	p_R	p_T	μ	σ	-
○	○	○	○	nan	nan	0.33
○	○	○	●	0.571	0.364	0.61
○	○	●	○	0.633	0.355	0.44
○	○	●	●	0.578	0.359	0.66
○	●	○	○	0.604	0.365	0.30
○	●	○	●	0.571	0.360	0.59
○	●	●	○	0.613	0.354	0.42
○	●	●	●	0.578	0.357	0.63
●	○	○	○	0.651	0.354	0.52
●	○	○	●	0.590	0.355	0.71
●	○	●	○	0.641	0.353	0.52
●	○	●	●	0.593	0.355	0.69
●	●	○	○	0.621	0.352	0.51
●	●	○	●	0.583	0.353	0.69
●	●	●	○	0.624	0.350	0.51
●	●	●	●	0.589	0.353	0.68

(a) Sensor permutations without occupancy on $\mathrm{H}_1^{\ominus}\mathrm{SE}(2)$.

Sensors				$\Delta[\Xi]/\sum I$		Accuracy
p_L	p_P	p_R	p_T	μ	σ	-
○	○	○	○	nan	nan	0.33
○	○	○	●	0.556	0.344	0.53
○	○	●	○	0.564	0.343	0.35
○	○	●	●	0.552	0.345	0.54
○	●	○	○	0.603	0.353	0.28
○	●	○	●	0.555	0.344	0.51
○	●	●	○	0.563	0.343	0.34
○	●	●	●	0.550	0.346	0.53
●	○	○	○	0.566	0.343	0.38
●	○	○	●	0.546	0.346	0.52
●	○	●	○	0.556	0.343	0.39
●	○	●	●	0.543	0.345	0.53
●	●	○	○	0.563	0.342	0.37
●	●	○	●	0.545	0.346	0.52
●	●	●	○	0.554	0.343	0.38
●	●	●	●	0.544	0.345	0.52

(b) Sensor permutations without occupancy on $\mathrm{H}_2^{\ominus}\mathrm{SE}(2)$.

Table D.47: Sensor permutations without occupancy on $\mathrm{H}_1^{\ominus}\mathrm{SE}(2)$ and $\mathrm{H}_2^{\ominus}\mathrm{SE}(2)$. Occupancy data disabled with $p_O[\circ]$. Intrinsic conflict (μ and σ) and classification accuracy.

Sensor	Accuracy ●		Accuracy ○		Difference ● − ○	
-	μ	σ	μ	σ	μ	σ
p_L	0.604	0.089	0.497	0.133	0.106	0.160
p_P	0.541	0.126	0.560	0.124	-0.019	0.176
p_R	0.569	0.102	0.532	0.142	0.036	0.175
p_T	0.657	0.040	0.444	0.083	0.214	0.092

(a) Sensor permutations without occupancy on $\mathrm{H}_1^{\ominus}\mathrm{SE}(2)$.

Sensor	Accuracy ●		Accuracy ○		Difference ● − ○	
-	μ	σ	μ	σ	μ	σ
p_L	0.451	0.071	0.426	0.103	0.025	0.126
p_P	0.431	0.093	0.446	0.086	-0.015	0.126
p_R	0.448	0.084	0.430	0.094	0.017	0.126
p_T	0.525	0.009	0.353	0.034	0.172	0.035

(b) Sensor permutations without occupancy on $\mathrm{H}_2^{\ominus}\mathrm{SE}(2)$.

Table D.48: Sensor permutations without occupancy on $\mathrm{H}_1^{\ominus}\mathrm{SE}(2)$ and $\mathrm{H}_2^{\ominus}\mathrm{SE}(2)$. Classification accuracy evaluated per sensor. Classification impact for each sensor calculated as difference between enabled ● and disabled ○ cases. Occupancy data disabled with $p_O[\circ]$.

D.10 Scenario VIa : Holonomic Optimization on SE(2)

D.10.1 Non-Noisy Optimization parameterized with SO(n) on $H_3^{\oplus}SE(2)$

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\circledcirc_*]^{\oplus}$
H_0	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 0.924 & -0.383 \\ 0.000 & 0.383 & 0.924 \end{bmatrix}$	$\circledcirc$ [0.38, 0.92, 1.00] $\circledcirc_*$ [0.27, 0.65, 0.71]	
I_0	$\begin{bmatrix} 0.617 & 0.782 & -0.090 \\ 0.736 & -0.533 & 0.417 \\ 0.278 & -0.324 & -0.904 \end{bmatrix}$	$\circledcirc$ [0.43, 0.95, 0.96] $\circledcirc_*$ [0.30, 0.67, 0.68]	$\Delta[\Xi_u]$ 0.1970 $\Delta[\circledcirc_*]^{\oplus}$ 0.0285
I_1	$\begin{bmatrix} -0.698 & 0.664 & -0.269 \\ 0.716 & 0.641 & -0.275 \\ -0.010 & -0.385 & -0.923 \end{bmatrix}$	$\circledcirc$ [0.39, 0.92, 1.00] $\circledcirc_*$ [0.27, 0.65, 0.71]	$\Delta[\Xi_u]$ 0.1444 $\Delta[\circledcirc_*]^{\oplus}$ 0.0012
I_2	$\begin{bmatrix} -0.756 & -0.605 & 0.250 \\ 0.655 & -0.698 & 0.289 \\ -0.000 & 0.382 & 0.924 \end{bmatrix}$	$\circledcirc$ [0.38, 0.92, 1.00] $\circledcirc_*$ [0.27, 0.65, 0.71]	$\Delta[\Xi_u]$ 0.0997 $\Delta[\circledcirc_*]^{\oplus}$ 0.0004
I_3	$\begin{bmatrix} -0.851 & 0.488 & -0.195 \\ 0.525 & 0.788 & -0.321 \\ -0.003 & -0.376 & -0.927 \end{bmatrix}$	$\circledcirc$ [0.38, 0.93, 1.00] $\circledcirc_*$ [0.27, 0.66, 0.71]	$\Delta[\Xi_u]$ 0.1261 $\Delta[\circledcirc_*]^{\oplus}$ 0.0034
I_4	$\begin{bmatrix} 0.341 & 0.870 & -0.357 \\ -0.940 & 0.316 & -0.129 \\ 0.001 & 0.380 & 0.925 \end{bmatrix}$	$\circledcirc$ [0.38, 0.93, 1.00] $\circledcirc_*$ [0.27, 0.65, 0.71]	$\Delta[\Xi_u]$ 0.1049 $\Delta[\circledcirc_*]^{\oplus}$ 0.0014

(a) Hypothesis H_0 for $n_\theta = 10$.

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\circledcirc_*]^{\oplus}$
H_0	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 0.924 & -0.383 \\ 0.000 & 0.383 & 0.924 \end{bmatrix}$	$\circledcirc$ [0.38, 0.92, 1.00] $\circledcirc_*$ [0.27, 0.65, 0.71]	
I_0	$\begin{bmatrix} -0.948 & -0.295 & 0.121 \\ -0.319 & 0.873 & -0.368 \\ 0.002 & -0.387 & -0.922 \end{bmatrix}$	$\circledcirc$ [0.39, 0.92, 1.00] $\circledcirc_*$ [0.27, 0.65, 0.71]	$\Delta[\Xi_u]$ 0.1149 $\Delta[\circledcirc_*]^{\oplus}$ 0.0023
I_1	$\begin{bmatrix} 0.709 & -0.652 & 0.270 \\ -0.706 & -0.655 & 0.271 \\ 0.000 & -0.383 & -0.924 \end{bmatrix}$	$\circledcirc$ [0.38, 0.92, 1.00] $\circledcirc_*$ [0.27, 0.65, 0.71]	$\Delta[\Xi_u]$ 0.1420 $\Delta[\circledcirc_*]^{\oplus}$ 0.0001
I_2	$\begin{bmatrix} -0.868 & -0.458 & 0.190 \\ -0.496 & 0.802 & -0.333 \\ 0.000 & -0.383 & -0.924 \end{bmatrix}$	$\circledcirc$ [0.38, 0.92, 1.00] $\circledcirc_*$ [0.27, 0.65, 0.71]	$\Delta[\Xi_u]$ 0.0996 $\Delta[\circledcirc_*]^{\oplus}$ 0.0001
I_3	$\begin{bmatrix} -0.696 & 0.664 & -0.273 \\ 0.718 & 0.643 & -0.267 \\ -0.001 & -0.382 & -0.924 \end{bmatrix}$	$\circledcirc$ [0.38, 0.92, 1.00] $\circledcirc_*$ [0.27, 0.65, 0.71]	$\Delta[\Xi_u]$ 0.1097 $\Delta[\circledcirc_*]^{\oplus}$ 0.0003
I_4	$\begin{bmatrix} 0.363 & 0.863 & -0.351 \\ -0.932 & 0.338 & -0.133 \\ 0.004 & 0.375 & 0.927 \end{bmatrix}$	$\circledcirc$ [0.38, 0.93, 1.00] $\circledcirc_*$ [0.27, 0.66, 0.71]	$\Delta[\Xi_u]$ 0.1054 $\Delta[\circledcirc_*]^{\oplus}$ 0.0037

(b) Hypothesis H_0 for $n_\theta = 20$.

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\circledcirc_*]^{\oplus}$
H_1	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 0.707 & -0.707 \\ 0.000 & 0.707 & 0.707 \end{bmatrix}$	$\circledcirc$ [0.71, 0.71, 1.00] $\circledcirc_*$ [0.50, 0.50, 0.71]	
I_0	$\begin{bmatrix} -0.999 & -0.023 & 0.033 \\ -0.040 & 0.709 & -0.704 \\ -0.007 & -0.705 & -0.709 \end{bmatrix}$	$\circledcirc$ [0.70, 0.71, 1.00] $\circledcirc_*$ [0.50, 0.50, 0.71]	$\Delta[\Xi_u]$ 0.1141 $\Delta[\circledcirc_*]^{\oplus}$ 0.0014
I_1	$\begin{bmatrix} 0.749 & 0.611 & -0.257 \\ -0.505 & 0.274 & -0.818 \\ -0.429 & 0.743 & 0.514 \end{bmatrix}$	$\circledcirc$ [0.86, 0.67, 0.90] $\circledcirc_*$ [0.61, 0.47, 0.64]	$\Delta[\Xi_u]$ 0.2364 $\Delta[\circledcirc_*]^{\oplus}$ 0.0824
I_2	$\begin{bmatrix} -1.000 & -0.006 & 0.007 \\ 0.009 & -0.710 & 0.704 \\ 0.000 & 0.704 & 0.710 \end{bmatrix}$	$\circledcirc$ [0.70, 0.71, 1.00] $\circledcirc_*$ [0.50, 0.50, 0.71]	$\Delta[\Xi_u]$ 0.1241 $\Delta[\circledcirc_*]^{\oplus}$ 0.0017
I_3	$\begin{bmatrix} -0.794 & 0.431 & -0.429 \\ 0.608 & 0.561 & -0.561 \\ -0.001 & -0.707 & -0.707 \end{bmatrix}$	$\circledcirc$ [0.71, 0.71, 1.00] $\circledcirc_*$ [0.50, 0.50, 0.71]	$\Delta[\Xi_u]$ 0.1470 $\Delta[\circledcirc_*]^{\oplus}$ 0.0002
I_4	$\begin{bmatrix} 0.853 & 0.370 & -0.368 \\ -0.522 & 0.607 & -0.599 \\ 0.001 & 0.703 & 0.711 \end{bmatrix}$	$\circledcirc$ [0.70, 0.71, 1.00] $\circledcirc_*$ [0.50, 0.50, 0.71]	$\Delta[\Xi_u]$ 0.1046 $\Delta[\circledcirc_*]^{\oplus}$ 0.0025

(c) Hypothesis H_1 for $n_\theta = 10$.

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\circledcirc_*]^{\oplus}$
H_1	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 0.707 & -0.707 \\ 0.000 & 0.707 & 0.707 \end{bmatrix}$	$\circledcirc$ [0.71, 0.71, 1.00] $\circledcirc_*$ [0.50, 0.50, 0.71]	
I_0	$\begin{bmatrix} -0.751 & -0.468 & 0.466 \\ -0.660 & 0.530 & -0.532 \\ 0.002 & -0.708 & -0.707 \end{bmatrix}$	$\circledcirc$ [0.71, 0.71, 1.00] $\circledcirc_*$ [0.50, 0.50, 0.71]	$\Delta[\Xi_u]$ 0.1057 $\Delta[\circledcirc_*]^{\oplus}$ 0.0003
I_1	$\begin{bmatrix} 0.534 & -0.594 & 0.602 \\ -0.846 & -0.376 & 0.379 \\ 0.002 & -0.712 & -0.703 \end{bmatrix}$	$\circledcirc$ [0.71, 0.70, 1.00] $\circledcirc_*$ [0.50, 0.50, 0.71]	$\Delta[\Xi_u]$ 0.1213 $\Delta[\circledcirc_*]^{\oplus}$ 0.0029
I_2	$\begin{bmatrix} -0.956 & -0.205 & 0.209 \\ 0.293 & -0.676 & 0.676 \\ 0.003 & 0.708 & 0.707 \end{bmatrix}$	$\circledcirc$ [0.71, 0.71, 1.00] $\circledcirc_*$ [0.50, 0.50, 0.71]	$\Delta[\Xi_u]$ 0.1221 $\Delta[\circledcirc_*]^{\oplus}$ 0.0003
I_3	$\begin{bmatrix} 0.250 & 0.687 & -0.682 \\ -0.968 & 0.181 & -0.173 \\ 0.005 & 0.704 & 0.710 \end{bmatrix}$	$\circledcirc$ [0.70, 0.71, 1.00] $\circledcirc_*$ [0.50, 0.50, 0.71]	$\Delta[\Xi_u]$ 0.1545 $\Delta[\circledcirc_*]^{\oplus}$ 0.0020
I_4	$\begin{bmatrix} -0.319 & 0.674 & -0.666 \\ 0.948 & 0.230 & -0.222 \\ 0.003 & -0.702 & -0.712 \end{bmatrix}$	$\circledcirc$ [0.70, 0.71, 1.00] $\circledcirc_*$ [0.50, 0.50, 0.71]	$\Delta[\Xi_u]$ 0.1046 $\Delta[\circledcirc_*]^{\oplus}$ 0.0034

(d) Hypothesis H_1 for $n_\theta = 20$.

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\circledcirc_*]^{\oplus}$
H_2	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 1.000 & 0.000 \\ 0.000 & 0.000 & 1.000 \end{bmatrix}$	$\circledcirc$ [0.00, 1.00, 1.00] $\circledcirc_*$ [0.00, 0.71, 0.71]	
I_0	$\begin{bmatrix} 0.555 & 0.832 & -0.002 \\ 0.832 & -0.555 & 0.009 \\ 0.007 & -0.007 & -1.000 \end{bmatrix}$	$\circledcirc$ [0.01, 1.00, 1.00] $\circledcirc_*$ [0.01, 0.71, 0.71]	$\Delta[\Xi_u]$ 0.1112 $\Delta[\circledcirc_*]^{\oplus}$ 0.0044
I_1	$\begin{bmatrix} -0.136 & -0.991 & 0.000 \\ -0.991 & 0.136 & -0.001 \\ 0.000 & -0.000 & -1.000 \end{bmatrix}$	$\circledcirc$ [0.00, 1.00, 1.00] $\circledcirc_*$ [0.00, 0.71, 0.71]	$\Delta[\Xi_u]$ 0.1419 $\Delta[\circledcirc_*]^{\oplus}$ 0.0003
I_2	$\begin{bmatrix} -0.225 & 0.974 & -0.005 \\ 0.974 & 0.225 & 0.000 \\ 0.001 & -0.005 & -1.000 \end{bmatrix}$	$\circledcirc$ [0.01, 1.00, 1.00] $\circledcirc_*$ [0.00, 0.71, 0.71]	$\Delta[\Xi_u]$ 0.1316 $\Delta[\circledcirc_*]^{\oplus}$ 0.0024
I_3	$\begin{bmatrix} -0.957 & -0.290 & 0.002 \\ 0.290 & -0.957 & -0.002 \\ 0.002 & -0.001 & 1.000 \end{bmatrix}$	$\circledcirc$ [0.00, 1.00, 1.00] $\circledcirc_*$ [0.00, 0.71, 0.71]	$\Delta[\Xi_u]$ 0.1050 $\Delta[\circledcirc_*]^{\oplus}$ 0.0012
I_4	$\begin{bmatrix} -0.137 & 0.991 & 0.001 \\ 0.991 & 0.137 & -0.002 \\ -0.002 & 0.001 & -1.000 \end{bmatrix}$	$\circledcirc$ [0.00, 1.00, 1.00] $\circledcirc_*$ [0.00, 0.71, 0.71]	$\Delta[\Xi_u]$ 0.1025 $\Delta[\circledcirc_*]^{\oplus}$ 0.0011

(e) Hypothesis H_2 for $n_\theta = 10$.

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\circledcirc_*]^{\oplus}$
H_2	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 1.000 & 0.000 \\ 0.000 & 0.000 & 1.000 \end{bmatrix}$	$\circledcirc$ [0.00, 1.00, 1.00] $\circledcirc_*$ [0.00, 0.71, 0.71]	
I_0	$\begin{bmatrix} 0.484 & 0.875 & -0.000 \\ 0.875 & -0.484 & 0.002 \\ 0.001 & -0.001 & -1.000 \end{bmatrix}$	$\circledcirc$ [0.00, 1.00, 1.00] $\circledcirc_*$ [0.00, 0.71, 0.71]	$\Delta[\Xi_u]$ 0.0878 $\Delta[\circledcirc_*]^{\oplus}$ 0.0007
I_1	$\begin{bmatrix} -0.613 & 0.790 & 0.005 \\ 0.790 & 0.613 & 0.003 \\ -0.000 & 0.006 & -1.000 \end{bmatrix}$	$\circledcirc$ [0.01, 1.00, 1.00] $\circledcirc_*$ [0.00, 0.71, 0.71]	$\Delta[\Xi_u]$ 0.1528 $\Delta[\circledcirc_*]^{\oplus}$ 0.0027
I_2	$\begin{bmatrix} -0.223 & 0.975 & -0.002 \\ 0.975 & 0.223 & 0.001 \\ 0.001 & -0.001 & -1.000 \end{bmatrix}$	$\circledcirc$ [0.00, 1.00, 1.00] $\circledcirc_*$ [0.00, 0.71, 0.71]	$\Delta[\Xi_u]$ 0.1250 $\Delta[\circledcirc_*]^{\oplus}$ 0.0009
I_3	$\begin{bmatrix} -0.669 & -0.743 & -0.000 \\ 0.743 & -0.669 & -0.004 \\ 0.003 & -0.003 & 1.000 \end{bmatrix}$	$\circledcirc$ [0.00, 1.00, 1.00] $\circledcirc_*$ [0.00, 0.71, 0.71]	$\Delta[\Xi_u]$ 0.1018 $\Delta[\circledcirc_*]^{\oplus}$ 0.0019
I_4	$\begin{bmatrix} -0.782 & -0.623 & 0.000 \\ -0.623 & 0.782 & 0.001 \\ -0.001 & 0.000 & -1.000 \end{bmatrix}$	$\circledcirc$ [0.00, 1.00, 1.00] $\circledcirc_*$ [0.00, 0.71, 0.71]	$\Delta[\Xi_u]$ 0.1022 $\Delta[\circledcirc_*]^{\oplus}$ 0.0004

(f) Hypothesis H_2 for $n_\theta = 20$.

Table D.49: Non-noisy optimization parameterized with SO(n) on $H_3^{\oplus}SE(2)$. True projection matrix and optimization results per iteration I_0 to I_4. Commutativity vector, unnormalized intrinsic conflict and commutativity delta.

Id	Hypothesis, $\Delta[\circledcirc_*]^{\oplus}$		
	H_0	H_1	H_2
I_0	0.0285	0.0014	0.0044
I_1	0.0012	0.0824	0.0003
I_2	0.0004	0.0017	0.0024
I_3	0.0034	0.0002	0.0012
I_4	0.0014	0.0025	0.0011
μ	0.0070	0.0176	0.0019
σ	0.0108	0.0324	0.0014
$\mu = 0.0088$, $\sigma = 0.0215$			

(a) Evaluation for $n_\theta = 10$.

Id	Hypothesis, $\Delta[\circledcirc_*]^{\oplus}$		
	H_0	H_1	H_2
I_0	0.0023	0.0003	0.0007
I_1	0.0001	0.0029	0.0027
I_2	0.0001	0.0003	0.0009
I_3	0.0003	0.0020	0.0019
I_4	0.0037	0.0034	0.0004
μ	0.0013	0.0018	0.0013
σ	0.0015	0.0013	0.0009
$\mu = 0.0015$, $\sigma = 0.0013$			

(b) Evaluation for $n_\theta = 20$.

Table D.50: Non-noisy optimization parameterized with SO(n) on $H_3^{\oplus}SE(2)$. Commutativity delta $\Delta[\circledcirc_*]^{\oplus}$ per iteration I_0 to I_4. Mean and standard deviation of $\Delta[\circledcirc_*]^{\oplus}$ per hypothesis H_0 to H_2 and over all hypotheses.

D.10.2 Non-Noisy Optimization parameterized with GL(n,R) on $H_3^{\oplus}SE(2)$

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\textcircled{D}_*]^{\oplus}$
H_0	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 0.924 & -0.383 \\ 0.000 & 0.383 & 0.924 \end{bmatrix}$	$\textcircled{D}$ [0.38, 0.92, 1.00] $\textcircled{D}_*$ [0.27, 0.65, 0.71]	
I_0	$\begin{bmatrix} 3.241 & -0.592 & 0.384 \\ 1.526 & -0.206 & 0.640 \\ 0.195 & -0.342 & 0.418 \end{bmatrix}$	$\textcircled{D}$ [1.10, 1.38, 0.18] $\textcircled{D}_*$ [0.62, 0.78, 0.10]	$\Delta[\Xi_u]$ 0.2137 $\Delta[\textcircled{D}_*]^{\oplus}$ 0.4638
I_1	$\begin{bmatrix} 0.952 & -0.652 & 0.111 \\ 0.569 & 1.086 & -0.220 \\ 0.033 & 0.254 & 0.974 \end{bmatrix}$	$\textcircled{D}$ [0.29, 1.08, 1.30] $\textcircled{D}_*$ [0.17, 0.63, 0.76]	$\Delta[\Xi_u]$ 0.2038 $\Delta[\textcircled{D}_*]^{\oplus}$ 0.0747
I_2	$\begin{bmatrix} 1.288 & 2.342 & -0.185 \\ 1.227 & 3.563 & -0.262 \\ 0.349 & -0.424 & 0.439 \end{bmatrix}$	$\textcircled{D}$ [2.23, 0.89, 1.74] $\textcircled{D}_*$ [0.75, 0.30, 0.59]	$\Delta[\Xi_u]$ 0.2039 $\Delta[\textcircled{D}_*]^{\oplus}$ 0.3940
I_3	$\begin{bmatrix} 0.240 & -0.001 & -0.281 \\ 0.287 & 0.376 & -0.407 \\ -0.072 & 0.008 & 0.502 \end{bmatrix}$	$\textcircled{D}$ [0.03, 0.15, 0.19] $\textcircled{D}_*$ [0.12, 0.62, 0.78]	$\Delta[\Xi_u]$ 0.2232 $\Delta[\textcircled{D}_*]^{\oplus}$ 0.1089
I_4	$\begin{bmatrix} 0.609 & 0.486 & 0.295 \\ 1.629 & 3.556 & 0.213 \\ -0.050 & -0.084 & 0.249 \end{bmatrix}$	$\textcircled{D}$ [0.05, 0.45, 0.92] $\textcircled{D}_*$ [0.05, 0.44, 0.90]	$\Delta[\Xi_u]$ 0.1823 $\Delta[\textcircled{D}_*]^{\oplus}$ 0.2322

(a) Hypothesis H_0 for $n_\theta = 10$.

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\textcircled{D}_*]^{\oplus}$
H_0	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 0.924 & -0.383 \\ 0.000 & 0.383 & 0.924 \end{bmatrix}$	$\textcircled{D}$ [0.38, 0.92, 1.00] $\textcircled{D}_*$ [0.27, 0.65, 0.71]	
I_0	$\begin{bmatrix} 3.682 & 0.036 & -0.414 \\ 1.573 & -0.001 & 0.235 \\ 0.048 & -0.256 & 0.279 \end{bmatrix}$	$\textcircled{D}$ [1.03, 1.13, 0.11] $\textcircled{D}_*$ [0.67, 0.74, 0.07]	$\Delta[\Xi_u]$ 0.2062 $\Delta[\textcircled{D}_*]^{\oplus}$ 0.4921
I_1	$\begin{bmatrix} 0.077 & -0.117 & -0.306 \\ -0.030 & 0.025 & -0.623 \\ 0.177 & 0.352 & -0.007 \end{bmatrix}$	$\textcircled{D}$ [0.05, 0.12, 0.24] $\textcircled{D}_*$ [0.18, 0.44, 0.88]	$\Delta[\Xi_u]$ 0.2257 $\Delta[\textcircled{D}_*]^{\oplus}$ 0.1843
I_2	$\begin{bmatrix} 3.014 & -3.022 & 1.043 \\ -1.870 & 4.130 & -2.301 \\ -0.186 & 0.311 & 0.166 \end{bmatrix}$	$\textcircled{D}$ [0.42, 1.01, 1.63] $\textcircled{D}_*$ [0.21, 0.52, 0.83]	$\Delta[\Xi_u]$ 0.1763 $\Delta[\textcircled{D}_*]^{\oplus}$ 0.1223
I_3	$\begin{bmatrix} 0.231 & -0.074 & -0.032 \\ 0.073 & 0.265 & 0.067 \\ 0.006 & 0.385 & 0.908 \end{bmatrix}$	$\textcircled{D}$ [0.09, 0.22, 0.22] $\textcircled{D}_*$ [0.29, 0.67, 0.68]	$\Delta[\Xi_u]$ 0.1493 $\Delta[\textcircled{D}_*]^{\oplus}$ 0.0236
I_4	$\begin{bmatrix} 1.062 & 0.843 & 0.217 \\ 0.983 & 1.368 & -0.206 \\ -0.061 & 0.072 & 0.265 \end{bmatrix}$	$\textcircled{D}$ [0.20, 0.39, 0.43] $\textcircled{D}_*$ [0.33, 0.63, 0.70]	$\Delta[\Xi_u]$ 0.1783 $\Delta[\textcircled{D}_*]^{\oplus}$ 0.0395

(b) Hypothesis H_0 for $n_\theta = 20$.

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\textcircled{D}_*]^{\oplus}$
H_1	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 0.707 & -0.707 \\ 0.000 & 0.707 & 0.707 \end{bmatrix}$	$\textcircled{D}$ [0.71, 0.71, 1.00] $\textcircled{D}_*$ [0.50, 0.50, 0.71]	
I_0	$\begin{bmatrix} 3.870 & -0.686 & 1.061 \\ 2.406 & -0.024 & 1.024 \\ 0.137 & -0.774 & 0.266 \end{bmatrix}$	$\textcircled{D}$ [3.45, 1.01, 1.01] $\textcircled{D}_*$ [0.92, 0.27, 0.27]	$\Delta[\Xi_u]$ 0.2157 $\Delta[\textcircled{D}_*]^{\oplus}$ 0.4208
I_1	$\begin{bmatrix} 0.935 & 0.598 & 0.324 \\ -0.766 & 0.574 & -0.078 \\ 0.200 & 0.610 & 0.898 \end{bmatrix}$	$\textcircled{D}$ [0.74, 1.03, 0.66] $\textcircled{D}_*$ [0.52, 0.72, 0.46]	$\Delta[\Xi_u]$ 0.2355 $\Delta[\textcircled{D}_*]^{\oplus}$ 0.2112
I_2	$\begin{bmatrix} 1.077 & 1.097 & -0.155 \\ 0.798 & 2.332 & -0.664 \\ 0.192 & -0.883 & 0.557 \end{bmatrix}$	$\textcircled{D}$ [1.64, 0.85, 0.86] $\textcircled{D}_*$ [0.80, 0.42, 0.42]	$\Delta[\Xi_u]$ 0.2644 $\Delta[\textcircled{D}_*]^{\oplus}$ 0.2728
I_3	$\begin{bmatrix} 0.472 & 0.216 & 0.425 \\ 0.534 & 0.519 & 0.444 \\ -0.233 & -0.029 & 0.226 \end{bmatrix}$	$\textcircled{D}$ [0.11, 0.30, 0.14] $\textcircled{D}_*$ [0.32, 0.86, 0.41]	$\Delta[\Xi_u]$ 0.2572 $\Delta[\textcircled{D}_*]^{\oplus}$ 0.3238
I_4	$\begin{bmatrix} 0.687 & 0.649 & 0.233 \\ 1.685 & 3.390 & -0.291 \\ -0.114 & 0.017 & 0.199 \end{bmatrix}$	$\textcircled{D}$ [0.42, 0.34, 0.69] $\textcircled{D}_*$ [0.48, 0.39, 0.78]	$\Delta[\Xi_u]$ 0.2040 $\Delta[\textcircled{D}_*]^{\oplus}$ 0.0864

(c) Hypothesis H_1 for $n_\theta = 10$.

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\textcircled{D}_*]^{\oplus}$
H_1	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 0.707 & -0.707 \\ 0.000 & 0.707 & 0.707 \end{bmatrix}$	$\textcircled{D}$ [0.71, 0.71, 1.00] $\textcircled{D}_*$ [0.50, 0.50, 0.71]	
I_0	$\begin{bmatrix} 4.376 & -1.077 & 0.391 \\ 2.634 & -0.166 & 0.876 \\ 0.083 & -0.754 & 0.004 \end{bmatrix}$	$\textcircled{D}$ [3.77, 0.06, 0.72] $\textcircled{D}_*$ [0.98, 0.02, 0.19]	$\Delta[\Xi_u]$ 0.2074 $\Delta[\textcircled{D}_*]^{\oplus}$ 0.5643
I_1	$\begin{bmatrix} 1.065 & 0.307 & 0.670 \\ -0.138 & 1.167 & -0.060 \\ 0.139 & 0.447 & 0.741 \end{bmatrix}$	$\textcircled{D}$ [0.49, 0.70, 0.89] $\textcircled{D}_*$ [0.39, 0.57, 0.72]	$\Delta[\Xi_u]$ 0.2318 $\Delta[\textcircled{D}_*]^{\oplus}$ 0.0803
I_2	$\begin{bmatrix} 2.750 & -2.567 & 0.363 \\ -1.670 & 3.876 & -2.132 \\ 0.098 & 0.107 & 0.311 \end{bmatrix}$	$\textcircled{D}$ [0.78, 0.88, 1.66] $\textcircled{D}_*$ [0.38, 0.43, 0.82]	$\Delta[\Xi_u]$ 0.2179 $\Delta[\textcircled{D}_*]^{\oplus}$ 0.1110
I_3	$\begin{bmatrix} 0.310 & 0.315 & 0.574 \\ 0.500 & 0.284 & -0.150 \\ -0.498 & 0.103 & 0.133 \end{bmatrix}$	$\textcircled{D}$ [0.27, 0.33, 0.06] $\textcircled{D}_*$ [0.63, 0.76, 0.13]	$\Delta[\Xi_u]$ 0.2571 $\Delta[\textcircled{D}_*]^{\oplus}$ 0.4197
I_4	$\begin{bmatrix} 0.485 & 0.735 & 0.081 \\ -0.213 & 0.489 & -0.518 \\ -0.037 & 0.315 & 0.402 \end{bmatrix}$	$\textcircled{D}$ [0.19, 0.22, 0.45] $\textcircled{D}_*$ [0.35, 0.42, 0.84]	$\Delta[\Xi_u]$ 0.1930 $\Delta[\textcircled{D}_*]^{\oplus}$ 0.1387

(d) Hypothesis H_1 for $n_\theta = 20$.

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\textcircled{D}_*]^{\oplus}$
H_2	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 1.000 & 0.000 \\ 0.000 & 0.000 & 1.000 \end{bmatrix}$	$\textcircled{D}$ [0.00, 1.00, 1.00] $\textcircled{D}_*$ [0.00, 0.71, 0.71]	
I_0	$\begin{bmatrix} -0.057 & -0.137 & -0.778 \\ 0.230 & 2.242 & 3.747 \\ 0.307 & -0.363 & 0.238 \end{bmatrix}$	$\textcircled{D}$ [0.77, 1.12, 1.92] $\textcircled{D}_*$ [0.33, 0.48, 0.82]	$\Delta[\Xi_u]$ 0.2435 $\Delta[\textcircled{D}_*]^{\oplus}$ 0.2672
I_1	$\begin{bmatrix} 4.428 & 1.342 & -3.425 \\ 1.602 & 0.835 & -2.087 \\ 0.427 & -0.432 & 3.401 \end{bmatrix}$	$\textcircled{D}$ [2.70, 17.70, 3.64] $\textcircled{D}_*$ [0.15, 0.97, 0.20]	$\Delta[\Xi_u]$ 0.2766 $\Delta[\textcircled{D}_*]^{\oplus}$ 0.3813
I_2	$\begin{bmatrix} 3.482 & 5.620 & -0.968 \\ 2.548 & 5.696 & -1.319 \\ 0.959 & 0.513 & 0.313 \end{bmatrix}$	$\textcircled{D}$ [5.50, 2.89, 3.34] $\textcircled{D}_*$ [0.78, 0.41, 0.47]	$\Delta[\Xi_u]$ 0.2893 $\Delta[\textcircled{D}_*]^{\oplus}$ 0.5705
I_3	$\begin{bmatrix} 0.268 & 0.019 & -0.058 \\ 0.421 & 0.467 & -0.206 \\ -0.150 & -0.066 & 0.353 \end{bmatrix}$	$\textcircled{D}$ [0.04, 0.15, 0.15] $\textcircled{D}_*$ [0.21, 0.68, 0.70]	$\Delta[\Xi_u]$ 0.1962 $\Delta[\textcircled{D}_*]^{\oplus}$ 0.1338
I_4	$\begin{bmatrix} 0.606 & 0.002 & 0.447 \\ 1.346 & 1.971 & 0.529 \\ -0.079 & -0.033 & 0.344 \end{bmatrix}$	$\textcircled{D}$ [0.11, 0.56, 0.70] $\textcircled{D}_*$ [0.13, 0.62, 0.77]	$\Delta[\Xi_u]$ 0.2189 $\Delta[\textcircled{D}_*]^{\oplus}$ 0.1051

(e) Hypothesis H_2 for $n_\theta = 10$.

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\textcircled{D}_*]^{\oplus}$
H_2	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 1.000 & 0.000 \\ 0.000 & 0.000 & 1.000 \end{bmatrix}$	$\textcircled{D}$ [0.00, 1.00, 1.00] $\textcircled{D}_*$ [0.00, 0.71, 0.71]	
I_0	$\begin{bmatrix} 0.169 & -0.302 & -1.001 \\ -0.571 & 2.625 & 2.413 \\ 0.400 & -0.734 & 0.697 \end{bmatrix}$	$\textcircled{D}$ [0.63, 1.46, 3.72] $\textcircled{D}_*$ [0.16, 0.36, 0.92]	$\Delta[\Xi_u]$ 0.2345 $\Delta[\textcircled{D}_*]^{\oplus}$ 0.2797
I_1	$\begin{bmatrix} 1.038 & 0.829 & 0.153 \\ 2.404 & 3.120 & -0.016 \\ 0.152 & -0.359 & 0.240 \end{bmatrix}$	$\textcircled{D}$ [1.43, 0.62, 0.79] $\textcircled{D}_*$ [0.82, 0.36, 0.45]	$\Delta[\Xi_u]$ 0.2414 $\Delta[\textcircled{D}_*]^{\oplus}$ 0.6129
I_2	$\begin{bmatrix} 1.561 & -2.458 & 0.610 \\ -0.562 & 3.786 & -2.205 \\ 0.053 & -0.174 & 0.451 \end{bmatrix}$	$\textcircled{D}$ [0.17, 0.69, 1.66] $\textcircled{D}_*$ [0.10, 0.38, 0.92]	$\Delta[\Xi_u]$ 0.2682 $\Delta[\textcircled{D}_*]^{\oplus}$ 0.2578
I_3	$\begin{bmatrix} 0.345 & 0.267 & 0.046 \\ 0.165 & 0.370 & -0.113 \\ -0.069 & 0.017 & 0.382 \end{bmatrix}$	$\textcircled{D}$ [0.04, 0.15, 0.18] $\textcircled{D}_*$ [0.16, 0.63, 0.76]	$\Delta[\Xi_u]$ 0.1943 $\Delta[\textcircled{D}_*]^{\oplus}$ 0.1186
I_4	$\begin{bmatrix} 0.507 & -0.445 & 0.511 \\ -0.144 & 0.413 & -0.556 \\ -0.215 & 0.218 & 0.115 \end{bmatrix}$	$\textcircled{D}$ [0.06, 0.22, 0.23] $\textcircled{D}_*$ [0.18, 0.67, 0.72]	$\Delta[\Xi_u]$ 0.2093 $\Delta[\textcircled{D}_*]^{\oplus}$ 0.1195

(f) Hypothesis H_2 for $n_\theta = 20$.

Table D.51: Non-noisy optimization parameterized with GL(n,R) on $H_3^{\oplus}SE(2)$. True projection matrix and optimization results per iteration I_0 to I_4. Commutativity vector, unnormalized intrinsic conflict and commutativity delta.

Id	Hypothesis, $\Delta[\textcircled{D}_*]^{\oplus}$		
	H_0	H_1	H_2
I_0	0.4638	0.4208	0.2672
I_1	0.0747	0.2112	0.3813
I_2	0.3940	0.2728	0.5705
I_3	0.1089	0.3238	0.1338
I_4	0.2322	0.0864	0.1051
μ	0.2547	0.2630	0.2916
σ	0.1532	0.1119	0.1709
$\mu = 0.2698$, $\sigma = 0.1535$			

(a) Evaluation for $n_\theta = 10$.

Id	Hypothesis, $\Delta[\textcircled{D}_*]^{\oplus}$		
	H_0	H_1	H_2
I_0	0.4921	0.5643	0.2797
I_1	0.1843	0.0803	0.6129
I_2	0.1223	0.1110	0.2578
I_3	0.0236	0.4197	0.1186
I_4	0.0395	0.1387	0.1195
μ	0.1724	0.2628	0.2777
σ	0.1701	0.1935	0.1806
$\mu = 0.2376$, $\sigma = 0.1941$			

(b) Evaluation for $n_\theta = 20$.

Table D.52: Non-noisy optimization parameterized with GL(n,R) on $H_3^{\oplus}SE(2)$. Commutativity delta $\Delta[\textcircled{D}_*]^{\oplus}$ per iteration I_0 to I_4. Mean and standard deviation of $\Delta[\textcircled{D}_*]^{\oplus}$ per hypothesis H_0 to H_2 and over all hypotheses.

D.10.3 Noisy Optimization parameterized with SO(n) on $\mathrm{H}_3^{\oplus}\mathrm{SE}(2)$

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\textcircled{1}_*]^{\oplus}$
H_0	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 0.924 & -0.383 \\ 0.000 & 0.383 & 0.924 \end{bmatrix}$	$\textcircled{1}$ [0.38, 0.92, 1.00] $\textcircled{1}_*$ [0.27, 0.65, 0.71]	
I_0	$\begin{bmatrix} 0.524 & 0.783 & -0.336 \\ 0.852 & -0.473 & 0.225 \\ 0.017 & -0.404 & -0.914 \end{bmatrix}$	$\textcircled{1}$ [0.40, 0.91, 1.00] $\textcircled{1}_*$ [0.29, 0.65, 0.71]	$\Delta[\Xi_u]$ 0.2189 $\Delta[\textcircled{1}_*]^{\oplus}$ 0.0108
I_1	$\begin{bmatrix} -0.913 & -0.386 & 0.133 \\ -0.407 & 0.840 & -0.360 \\ 0.027 & -0.383 & -0.924 \end{bmatrix}$	$\textcircled{1}$ [0.38, 0.92, 1.00] $\textcircled{1}_*$ [0.27, 0.65, 0.71]	$\Delta[\Xi_u]$ 0.2182 $\Delta[\textcircled{1}_*]^{\oplus}$ 0.0004
I_2	$\begin{bmatrix} -0.995 & -0.081 & 0.058 \\ 0.097 & -0.927 & 0.362 \\ 0.024 & 0.366 & 0.930 \end{bmatrix}$	$\textcircled{1}$ [0.37, 0.93, 1.00] $\textcircled{1}_*$ [0.26, 0.66, 0.71]	$\Delta[\Xi_u]$ 0.2024 $\Delta[\textcircled{1}_*]^{\oplus}$ 0.0079
I_3	$\begin{bmatrix} -0.679 & 0.657 & -0.328 \\ 0.732 & 0.642 & -0.228 \\ 0.061 & -0.395 & -0.917 \end{bmatrix}$	$\textcircled{1}$ [0.40, 0.92, 1.00] $\textcircled{1}_*$ [0.28, 0.65, 0.71]	$\Delta[\Xi_u]$ 0.2360 $\Delta[\textcircled{1}_*]^{\oplus}$ 0.0079
I_4	$\begin{bmatrix} -0.221 & 0.889 & -0.401 \\ 0.975 & 0.190 & -0.115 \\ -0.026 & -0.416 & -0.909 \end{bmatrix}$	$\textcircled{1}$ [0.42, 0.91, 1.00] $\textcircled{1}_*$ [0.29, 0.64, 0.71]	$\Delta[\Xi_u]$ 0.1832 $\Delta[\textcircled{1}_*]^{\oplus}$ 0.0167

(a) Hypothesis H_0 for $n_\theta = 10$.

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\textcircled{1}_*]^{\oplus}$
H_0	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 0.924 & -0.383 \\ 0.000 & 0.383 & 0.924 \end{bmatrix}$	$\textcircled{1}$ [0.38, 0.92, 1.00] $\textcircled{1}_*$ [0.27, 0.65, 0.71]	
I_0	$\begin{bmatrix} 0.552 & 0.769 & -0.323 \\ 0.834 & -0.512 & 0.204 \\ -0.008 & -0.382 & -0.924 \end{bmatrix}$	$\textcircled{1}$ [0.38, 0.92, 1.00] $\textcircled{1}_*$ [0.27, 0.65, 0.71]	$\Delta[\Xi_u]$ 0.1982 $\Delta[\textcircled{1}_*]^{\oplus}$ 0.0002
I_1	$\begin{bmatrix} 0.480 & -0.802 & 0.356 \\ -0.877 & -0.454 & 0.158 \\ 0.035 & -0.388 & -0.921 \end{bmatrix}$	$\textcircled{1}$ [0.39, 0.92, 1.00] $\textcircled{1}_*$ [0.28, 0.65, 0.71]	$\Delta[\Xi_u]$ 0.2180 $\Delta[\textcircled{1}_*]^{\oplus}$ 0.0033
I_2	$\begin{bmatrix} -0.895 & -0.407 & 0.185 \\ -0.446 & 0.839 & -0.312 \\ -0.029 & -0.361 & -0.932 \end{bmatrix}$	$\textcircled{1}$ [0.36, 0.93, 1.00] $\textcircled{1}_*$ [0.26, 0.66, 0.71]	$\Delta[\Xi_u]$ 0.2056 $\Delta[\textcircled{1}_*]^{\oplus}$ 0.0099
I_3	$\begin{bmatrix} -0.993 & -0.119 & -0.012 \\ 0.105 & -0.913 & 0.394 \\ -0.058 & 0.390 & 0.919 \end{bmatrix}$	$\textcircled{1}$ [0.39, 0.92, 1.00] $\textcircled{1}_*$ [0.28, 0.65, 0.71]	$\Delta[\Xi_u]$ 0.2296 $\Delta[\textcircled{1}_*]^{\oplus}$ 0.0055
I_4	$\begin{bmatrix} 0.957 & -0.272 & 0.098 \\ 0.287 & 0.863 & -0.416 \\ 0.029 & 0.426 & 0.904 \end{bmatrix}$	$\textcircled{1}$ [0.43, 0.90, 1.00] $\textcircled{1}_*$ [0.30, 0.64, 0.71]	$\Delta[\Xi_u]$ 0.1857 $\Delta[\textcircled{1}_*]^{\oplus}$ 0.0218

(b) Hypothesis H_0 for $n_\theta = 20$.

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\textcircled{1}_*]^{\oplus}$
H_1	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 0.707 & -0.707 \\ 0.000 & 0.707 & 0.707 \end{bmatrix}$	$\textcircled{1}$ [0.71, 0.71, 1.00] $\textcircled{1}_*$ [0.50, 0.50, 0.71]	
I_0	$\begin{bmatrix} -0.753 & -0.460 & 0.470 \\ -0.658 & 0.544 & -0.521 \\ -0.016 & -0.701 & -0.713 \end{bmatrix}$	$\textcircled{1}$ [0.70, 0.71, 1.00] $\textcircled{1}_*$ [0.50, 0.50, 0.71]	$\Delta[\Xi_u]$ 0.1945 $\Delta[\textcircled{1}_*]^{\oplus}$ 0.0035
I_1	$\begin{bmatrix} 0.943 & 0.270 & -0.193 \\ -0.329 & 0.690 & -0.645 \\ -0.041 & 0.672 & 0.740 \end{bmatrix}$	$\textcircled{1}$ [0.67, 0.74, 1.00] $\textcircled{1}_*$ [0.48, 0.52, 0.71]	$\Delta[\Xi_u]$ 0.2003 $\Delta[\textcircled{1}_*]^{\oplus}$ 0.0215
I_2	$\begin{bmatrix} -1.000 & -0.003 & 0.025 \\ 0.019 & -0.716 & 0.698 \\ 0.016 & 0.698 & 0.716 \end{bmatrix}$	$\textcircled{1}$ [0.70, 0.72, 1.00] $\textcircled{1}_*$ [0.49, 0.51, 0.71]	$\Delta[\Xi_u]$ 0.2000 $\Delta[\textcircled{1}_*]^{\oplus}$ 0.0056
I_3	$\begin{bmatrix} -0.961 & 0.113 & -0.252 \\ 0.254 & 0.718 & -0.648 \\ 0.108 & -0.687 & -0.718 \end{bmatrix}$	$\textcircled{1}$ [0.70, 0.73, 0.99] $\textcircled{1}_*$ [0.49, 0.51, 0.70]	$\Delta[\Xi_u]$ 0.2392 $\Delta[\textcircled{1}_*]^{\oplus}$ 0.0105
I_4	$\begin{bmatrix} 0.248 & 0.686 & -0.684 \\ 0.968 & -0.197 & 0.154 \\ -0.029 & -0.701 & -0.713 \end{bmatrix}$	$\textcircled{1}$ [0.70, 0.71, 1.00] $\textcircled{1}_*$ [0.50, 0.50, 0.71]	$\Delta[\Xi_u]$ 0.1638 $\Delta[\textcircled{1}_*]^{\oplus}$ 0.0038

(c) Hypothesis H_1 for $n_\theta = 10$.

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\textcircled{1}_*]^{\oplus}$
H_1	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 0.707 & -0.707 \\ 0.000 & 0.707 & 0.707 \end{bmatrix}$	$\textcircled{1}$ [0.71, 0.71, 1.00] $\textcircled{1}_*$ [0.50, 0.50, 0.71]	
I_0	$\begin{bmatrix} -0.862 & -0.353 & 0.365 \\ -0.508 & 0.613 & -0.606 \\ -0.010 & -0.707 & -0.707 \end{bmatrix}$	$\textcircled{1}$ [0.71, 0.71, 1.00] $\textcircled{1}_*$ [0.50, 0.50, 0.71]	$\Delta[\Xi_u]$ 0.1940 $\Delta[\textcircled{1}_*]^{\oplus}$ 0.0001
I_1	$\begin{bmatrix} 0.798 & -0.422 & 0.431 \\ -0.601 & -0.614 & 0.512 \\ 0.048 & -0.667 & -0.744 \end{bmatrix}$	$\textcircled{1}$ [0.67, 0.75, 1.00] $\textcircled{1}_*$ [0.47, 0.53, 0.71]	$\Delta[\Xi_u]$ 0.1984 $\Delta[\textcircled{1}_*]^{\oplus}$ 0.0243
I_2	$\begin{bmatrix} 0.746 & -0.484 & 0.457 \\ 0.665 & 0.527 & -0.529 \\ 0.015 & 0.699 & 0.715 \end{bmatrix}$	$\textcircled{1}$ [0.70, 0.72, 1.00] $\textcircled{1}_*$ [0.49, 0.51, 0.71]	$\Delta[\Xi_u]$ 0.1991 $\Delta[\textcircled{1}_*]^{\oplus}$ 0.0052
I_3	$\begin{bmatrix} -0.658 & -0.593 & 0.465 \\ -0.747 & 0.434 & -0.504 \\ 0.097 & -0.679 & -0.728 \end{bmatrix}$	$\textcircled{1}$ [0.69, 0.73, 1.00] $\textcircled{1}_*$ [0.48, 0.52, 0.70]	$\Delta[\Xi_u]$ 0.2382 $\Delta[\textcircled{1}_*]^{\oplus}$ 0.0157
I_4	$\begin{bmatrix} 0.749 & 0.467 & -0.470 \\ -0.662 & 0.544 & -0.515 \\ 0.015 & 0.697 & 0.717 \end{bmatrix}$	$\textcircled{1}$ [0.70, 0.72, 1.00] $\textcircled{1}_*$ [0.49, 0.51, 0.71]	$\Delta[\Xi_u]$ 0.1625 $\Delta[\textcircled{1}_*]^{\oplus}$ 0.0066

(d) Hypothesis H_1 for $n_\theta = 20$.

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\textcircled{1}_*]^{\oplus}$
H_2	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 1.000 & 0.000 \\ 0.000 & 0.000 & 1.000 \end{bmatrix}$	$\textcircled{1}$ [0.00, 1.00, 1.00] $\textcircled{1}_*$ [0.00, 0.71, 0.71]	
I_0	$\begin{bmatrix} 0.599 & 0.801 & -0.009 \\ 0.801 & -0.599 & -0.008 \\ -0.012 & -0.002 & -1.000 \end{bmatrix}$	$\textcircled{1}$ [0.01, 1.00, 1.00] $\textcircled{1}_*$ [0.01, 0.71, 0.71]	$\Delta[\Xi_u]$ 0.1875 $\Delta[\textcircled{1}_*]^{\oplus}$ 0.0054
I_1	$\begin{bmatrix} 0.551 & -0.832 & -0.069 \\ -0.833 & -0.553 & 0.009 \\ -0.046 & 0.053 & -0.998 \end{bmatrix}$	$\textcircled{1}$ [0.07, 1.00, 1.00] $\textcircled{1}_*$ [0.05, 0.71, 0.71]	$\Delta[\Xi_u]$ 0.1873 $\Delta[\textcircled{1}_*]^{\oplus}$ 0.0315
I_2	$\begin{bmatrix} -0.164 & 0.986 & -0.036 \\ 0.986 & 0.163 & -0.024 \\ -0.018 & -0.040 & -0.999 \end{bmatrix}$	$\textcircled{1}$ [0.04, 1.00, 1.00] $\textcircled{1}_*$ [0.03, 0.71, 0.71]	$\Delta[\Xi_u]$ 0.2684 $\Delta[\textcircled{1}_*]^{\oplus}$ 0.0196
I_3	$\begin{bmatrix} -0.914 & -0.405 & 0.003 \\ 0.405 & -0.914 & -0.014 \\ 0.009 & -0.012 & 1.000 \end{bmatrix}$	$\textcircled{1}$ [0.01, 1.00, 1.00] $\textcircled{1}_*$ [0.01, 0.71, 0.71]	$\Delta[\Xi_u]$ 0.1727 $\Delta[\textcircled{1}_*]^{\oplus}$ 0.0066
I_4	$\begin{bmatrix} 0.839 & 0.544 & 0.022 \\ -0.544 & 0.839 & -0.009 \\ -0.023 & -0.004 & 1.000 \end{bmatrix}$	$\textcircled{1}$ [0.02, 1.00, 1.00] $\textcircled{1}_*$ [0.02, 0.71, 0.71]	$\Delta[\Xi_u]$ 0.1799 $\Delta[\textcircled{1}_*]^{\oplus}$ 0.0108

(e) Hypothesis H_2 for $n_\theta = 10$.

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\textcircled{1}_*]^{\oplus}$
H_2	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 1.000 & 0.000 \\ 0.000 & 0.000 & 1.000 \end{bmatrix}$	$\textcircled{1}$ [0.00, 1.00, 1.00] $\textcircled{1}_*$ [0.00, 0.71, 0.71]	
I_0	$\begin{bmatrix} 0.663 & 0.749 & 0.011 \\ 0.749 & -0.663 & -0.005 \\ 0.003 & 0.011 & -1.000 \end{bmatrix}$	$\textcircled{1}$ [0.01, 1.00, 1.00] $\textcircled{1}_*$ [0.01, 0.71, 0.71]	$\Delta[\Xi_u]$ 0.1844 $\Delta[\textcircled{1}_*]^{\oplus}$ 0.0052
I_1	$\begin{bmatrix} -0.356 & -0.932 & -0.075 \\ -0.934 & 0.352 & 0.052 \\ -0.022 & 0.088 & -0.996 \end{bmatrix}$	$\textcircled{1}$ [0.09, 1.00, 1.00] $\textcircled{1}_*$ [0.06, 0.70, 0.71]	$\Delta[\Xi_u]$ 0.1937 $\Delta[\textcircled{1}_*]^{\oplus}$ 0.0410
I_2	$\begin{bmatrix} -0.280 & 0.960 & -0.029 \\ 0.960 & 0.279 & -0.025 \\ -0.015 & -0.035 & -0.999 \end{bmatrix}$	$\textcircled{1}$ [0.04, 1.00, 1.00] $\textcircled{1}_*$ [0.03, 0.71, 0.71]	$\Delta[\Xi_u]$ 0.2625 $\Delta[\textcircled{1}_*]^{\oplus}$ 0.0173
I_3	$\begin{bmatrix} -0.986 & -0.165 & 0.018 \\ 0.165 & -0.986 & -0.011 \\ 0.019 & -0.008 & 1.000 \end{bmatrix}$	$\textcircled{1}$ [0.02, 1.00, 1.00] $\textcircled{1}_*$ [0.01, 0.71, 0.71]	$\Delta[\Xi_u]$ 0.1739 $\Delta[\textcircled{1}_*]^{\oplus}$ 0.0094
I_4	$\begin{bmatrix} -0.803 & -0.589 & -0.085 \\ -0.590 & 0.807 & -0.022 \\ 0.082 & 0.032 & -0.996 \end{bmatrix}$	$\textcircled{1}$ [0.09, 1.00, 1.00] $\textcircled{1}_*$ [0.06, 0.71, 0.70]	$\Delta[\Xi_u]$ 0.1688 $\Delta[\textcircled{1}_*]^{\oplus}$ 0.0396

(f) Hypothesis H_2 for $n_\theta = 20$.

Table D.53: Noisy optimization parameterized with SO(n) on $\mathrm{H}_3^{\oplus}\mathrm{SE}(2)$. Hypotheses H_0 to H_2. Sensor and actuator noise set to $\sigma_c, \sigma_p = 0.05$. True projection matrix and optimization results per iteration I_0 to I_4. Commutativity vector, unnormalized intrinsic conflict and commutativity delta.

Id	Hypothesis, $\Delta[\textcircled{1}_*]^{\oplus}$		
	H_0	H_1	H_2
I_0	0.0108	0.0035	0.0054
I_1	0.0004	0.0215	0.0315
I_2	0.0079	0.0056	0.0196
I_3	0.0079	0.0105	0.0066
I_4	0.0167	0.0038	0.0108
μ	0.0087	0.0090	0.0148
σ	0.0053	0.0067	0.0097
$\mu = 0.0108, \sigma = 0.0083$			

(a) Evaluation for $n_\theta = 10$.

Id	Hypothesis, $\Delta[\textcircled{1}_*]^{\oplus}$		
	H_0	H_1	H_2
I_0	0.0002	0.0001	0.0052
I_1	0.0033	0.0243	0.0410
I_2	0.0099	0.0052	0.0173
I_3	0.0055	0.0157	0.0094
I_4	0.0218	0.0066	0.0396
μ	0.0081	0.0104	0.0225
σ	0.0075	0.0086	0.0151
$\mu = 0.0137, \sigma = 0.0130$			

(b) Evaluation for $n_\theta = 20$.

Table D.54: Noisy optimization parameterized with SO(n) on $\mathrm{H}_3^{\oplus}\mathrm{SE}(2)$. Sensor and actuator noise set to $\sigma_c, \sigma_p = 0.05$. Commutativity delta $\Delta[\textcircled{1}_*]^{\oplus}$ per iteration I_0 to I_4. Mean and standard deviation of $\Delta[\textcircled{1}_*]^{\oplus}$ per hypothesis H_0 to H_2 and over all hypotheses.

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\textcircled{D}_*]^{\oplus}$
H_0	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 0.924 & -0.383 \\ 0.000 & 0.383 & 0.924 \end{bmatrix}$	$\textcircled{D}$ [0.38, 0.92, 1.00] $\textcircled{D}_*$ [0.27, 0.65, 0.71]	
I_0	$\begin{bmatrix} 0.994 & 0.103 & 0.027 \\ 0.081 & -0.897 & 0.434 \\ 0.068 & -0.429 & -0.901 \end{bmatrix}$	$\textcircled{D}$ [0.43, 0.90, 1.00] $\textcircled{D}_*$ [0.31, 0.64, 0.71]	$\Delta[\Xi_u]$ 0.2828 $\Delta[\textcircled{D}_*]^{\oplus}$ 0.0251
I_1	$\begin{bmatrix} -0.915 & -0.386 & 0.113 \\ -0.401 & 0.853 & -0.335 \\ 0.033 & -0.352 & -0.935 \end{bmatrix}$	$\textcircled{D}$ [0.35, 0.94, 1.00] $\textcircled{D}_*$ [0.25, 0.66, 0.71]	$\Delta[\Xi_u]$ 0.2764 $\Delta[\textcircled{D}_*]^{\oplus}$ 0.0143
I_2	$\begin{bmatrix} -0.747 & -0.613 & 0.258 \\ 0.665 & -0.687 & 0.293 \\ -0.002 & 0.390 & 0.921 \end{bmatrix}$	$\textcircled{D}$ [0.39, 0.92, 1.00] $\textcircled{D}_*$ [0.28, 0.65, 0.71]	$\Delta[\Xi_u]$ 0.2686 $\Delta[\textcircled{D}_*]^{\oplus}$ 0.0036
I_3	$\begin{bmatrix} -0.814 & 0.548 & -0.190 \\ 0.580 & 0.772 & -0.259 \\ 0.005 & -0.321 & -0.947 \end{bmatrix}$	$\textcircled{D}$ [0.32, 0.95, 1.00] $\textcircled{D}_*$ [0.23, 0.67, 0.71]	$\Delta[\Xi_u]$ 0.3049 $\Delta[\textcircled{D}_*]^{\oplus}$ 0.0297
I_4	$\begin{bmatrix} 0.856 & 0.462 & -0.231 \\ -0.514 & 0.804 & -0.300 \\ 0.047 & 0.376 & 0.926 \end{bmatrix}$	$\textcircled{D}$ [0.38, 0.93, 1.00] $\textcircled{D}_*$ [0.27, 0.66, 0.71]	$\Delta[\Xi_u]$ 0.2543 $\Delta[\textcircled{D}_*]^{\oplus}$ 0.0024

(a) Hypothesis H_0 for $n_\theta = 10$.

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\textcircled{D}_*]^{\oplus}$
H_0	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 0.924 & -0.383 \\ 0.000 & 0.383 & 0.924 \end{bmatrix}$	$\textcircled{D}$ [0.38, 0.92, 1.00] $\textcircled{D}_*$ [0.27, 0.65, 0.71]	
I_0	$\begin{bmatrix} 0.739 & -0.586 & 0.332 \\ 0.671 & 0.686 & -0.283 \\ -0.062 & 0.432 & 0.900 \end{bmatrix}$	$\textcircled{D}$ [0.44, 0.90, 1.00] $\textcircled{D}_*$ [0.31, 0.64, 0.71]	$\Delta[\Xi_u]$ 0.2820 $\Delta[\textcircled{D}_*]^{\oplus}$ 0.0261
I_1	$\begin{bmatrix} 0.791 & -0.564 & 0.238 \\ -0.611 & -0.747 & 0.263 \\ 0.030 & -0.354 & -0.935 \end{bmatrix}$	$\textcircled{D}$ [0.35, 0.94, 1.00] $\textcircled{D}_*$ [0.25, 0.66, 0.71]	$\Delta[\Xi_u]$ 0.2764 $\Delta[\textcircled{D}_*]^{\oplus}$ 0.0136
I_2	$\begin{bmatrix} -0.935 & -0.331 & 0.131 \\ -0.355 & 0.866 & -0.351 \\ 0.003 & -0.375 & -0.927 \end{bmatrix}$	$\textcircled{D}$ [0.37, 0.93, 1.00] $\textcircled{D}_*$ [0.26, 0.66, 0.71]	$\Delta[\Xi_u]$ 0.2702 $\Delta[\textcircled{D}_*]^{\oplus}$ 0.0039
I_3	$\begin{bmatrix} -0.721 & 0.653 & -0.232 \\ 0.693 & 0.682 & -0.234 \\ 0.005 & -0.329 & -0.944 \end{bmatrix}$	$\textcircled{D}$ [0.33, 0.94, 1.00] $\textcircled{D}_*$ [0.23, 0.67, 0.71]	$\Delta[\Xi_u]$ 0.3017 $\Delta[\textcircled{D}_*]^{\oplus}$ 0.0257
I_4	$\begin{bmatrix} -0.889 & 0.444 & -0.112 \\ 0.452 & 0.809 & -0.377 \\ -0.077 & -0.386 & -0.919 \end{bmatrix}$	$\textcircled{D}$ [0.39, 0.92, 1.00] $\textcircled{D}_*$ [0.28, 0.65, 0.71]	$\Delta[\Xi_u]$ 0.2511 $\Delta[\textcircled{D}_*]^{\oplus}$ 0.0051

(b) Hypothesis H_0 for $n_\theta = 20$.

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\textcircled{D}_*]^{\oplus}$
H_1	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 0.707 & -0.707 \\ 0.000 & 0.707 & 0.707 \end{bmatrix}$	$\textcircled{D}$ [0.71, 0.71, 1.00] $\textcircled{D}_*$ [0.50, 0.50, 0.71]	
I_0	$\begin{bmatrix} 0.987 & 0.127 & -0.098 \\ 0.158 & -0.667 & 0.728 \\ 0.027 & -0.734 & -0.679 \end{bmatrix}$	$\textcircled{D}$ [0.73, 0.68, 1.00] $\textcircled{D}_*$ [0.52, 0.48, 0.71]	$\Delta[\Xi_u]$ 0.2633 $\Delta[\textcircled{D}_*]^{\oplus}$ 0.0176
I_1	$\begin{bmatrix} -0.203 & -0.737 & 0.644 \\ 0.979 & -0.129 & 0.161 \\ -0.036 & 0.663 & 0.748 \end{bmatrix}$	$\textcircled{D}$ [0.66, 0.75, 1.00] $\textcircled{D}_*$ [0.47, 0.53, 0.71]	$\Delta[\Xi_u]$ 0.2451 $\Delta[\textcircled{D}_*]^{\oplus}$ 0.0269
I_2	$\begin{bmatrix} -0.993 & -0.093 & 0.067 \\ 0.113 & -0.721 & 0.684 \\ -0.015 & 0.687 & 0.727 \end{bmatrix}$	$\textcircled{D}$ [0.69, 0.73, 1.00] $\textcircled{D}_*$ [0.49, 0.51, 0.71]	$\Delta[\Xi_u]$ 0.2727 $\Delta[\textcircled{D}_*]^{\oplus}$ 0.0126
I_3	$\begin{bmatrix} 0.747 & 0.481 & -0.460 \\ -0.508 & 0.858 & 0.072 \\ 0.429 & 0.180 & 0.885 \end{bmatrix}$	$\textcircled{D}$ [0.47, 0.98, 0.90] $\textcircled{D}_*$ [0.33, 0.70, 0.64]	$\Delta[\Xi_u]$ 0.3292 $\Delta[\textcircled{D}_*]^{\oplus}$ 0.1714
I_4	$\begin{bmatrix} -0.138 & 0.716 & -0.684 \\ 0.990 & 0.076 & -0.121 \\ -0.035 & -0.694 & -0.720 \end{bmatrix}$	$\textcircled{D}$ [0.69, 0.72, 1.00] $\textcircled{D}_*$ [0.49, 0.51, 0.71]	$\Delta[\Xi_u]$ 0.2653 $\Delta[\textcircled{D}_*]^{\oplus}$ 0.0083

(c) Hypothesis H_1 for $n_\theta = 10$.

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\textcircled{D}_*]^{\oplus}$
H_1	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 0.707 & -0.707 \\ 0.000 & 0.707 & 0.707 \end{bmatrix}$	$\textcircled{D}$ [0.71, 0.71, 1.00] $\textcircled{D}_*$ [0.50, 0.50, 0.71]	
I_0	$\begin{bmatrix} 0.749 & 0.427 & -0.506 \\ 0.660 & -0.537 & 0.525 \\ -0.048 & -0.728 & -0.684 \end{bmatrix}$	$\textcircled{D}$ [0.73, 0.69, 1.00] $\textcircled{D}_*$ [0.52, 0.48, 0.71]	$\Delta[\Xi_u]$ 0.2557 $\Delta[\textcircled{D}_*]^{\oplus}$ 0.0139
I_1	$\begin{bmatrix} -0.023 & 0.723 & -0.691 \\ -0.998 & -0.053 & -0.023 \\ -0.053 & 0.689 & 0.723 \end{bmatrix}$	$\textcircled{D}$ [0.69, 0.72, 1.00] $\textcircled{D}_*$ [0.49, 0.51, 0.71]	$\Delta[\Xi_u]$ 0.2453 $\Delta[\textcircled{D}_*]^{\oplus}$ 0.0107
I_2	$\begin{bmatrix} -0.983 & -0.133 & 0.125 \\ 0.182 & -0.700 & 0.691 \\ -0.005 & 0.702 & 0.712 \end{bmatrix}$	$\textcircled{D}$ [0.70, 0.71, 1.00] $\textcircled{D}_*$ [0.50, 0.50, 0.71]	$\Delta[\Xi_u]$ 0.2623 $\Delta[\textcircled{D}_*]^{\oplus}$ 0.0034
I_3	$\begin{bmatrix} -0.242 & -0.486 & 0.840 \\ -0.951 & -0.053 & -0.305 \\ 0.193 & -0.872 & -0.449 \end{bmatrix}$	$\textcircled{D}$ [0.89, 0.49, 0.98] $\textcircled{D}_*$ [0.63, 0.35, 0.69]	$\Delta[\Xi_u]$ 0.2964 $\Delta[\textcircled{D}_*]^{\oplus}$ 0.1297
I_4	$\begin{bmatrix} 0.098 & 0.715 & -0.692 \\ 0.995 & -0.094 & 0.044 \\ -0.033 & -0.692 & -0.721 \end{bmatrix}$	$\textcircled{D}$ [0.69, 0.72, 1.00] $\textcircled{D}_*$ [0.49, 0.51, 0.71]	$\Delta[\Xi_u]$ 0.2629 $\Delta[\textcircled{D}_*]^{\oplus}$ 0.0091

(d) Hypothesis H_1 for $n_\theta = 20$.

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\textcircled{D}_*]^{\oplus}$
H_2	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 1.000 & 0.000 \\ 0.000 & 0.000 & 1.000 \end{bmatrix}$	$\textcircled{D}$ [0.00, 1.00, 1.00] $\textcircled{D}_*$ [0.00, 0.71, 0.71]	
I_0	$\begin{bmatrix} 0.648 & 0.762 & -0.026 \\ 0.762 & -0.648 & 0.014 \\ -0.006 & -0.028 & -1.000 \end{bmatrix}$	$\textcircled{D}$ [0.03, 1.00, 1.00] $\textcircled{D}_*$ [0.02, 0.71, 0.71]	$\Delta[\Xi_u]$ 0.3093 $\Delta[\textcircled{D}_*]^{\oplus}$ 0.0131
I_1	$\begin{bmatrix} -0.925 & 0.368 & 0.089 \\ 0.342 & 0.913 & -0.220 \\ -0.162 & -0.173 & -0.971 \end{bmatrix}$	$\textcircled{D}$ [0.24, 0.98, 0.99] $\textcircled{D}_*$ [0.17, 0.70, 0.70]	$\Delta[\Xi_u]$ 0.3854 $\Delta[\textcircled{D}_*]^{\oplus}$ 0.1072
I_2	$\begin{bmatrix} -0.249 & 0.968 & 0.019 \\ 0.967 & 0.250 & -0.045 \\ -0.049 & 0.007 & -0.999 \end{bmatrix}$	$\textcircled{D}$ [0.05, 1.00, 1.00] $\textcircled{D}_*$ [0.03, 0.71, 0.71]	$\Delta[\Xi_u]$ 0.3044 $\Delta[\textcircled{D}_*]^{\oplus}$ 0.0222
I_3	$\begin{bmatrix} -0.889 & -0.454 & 0.058 \\ 0.453 & -0.891 & -0.031 \\ 0.066 & -0.001 & 0.998 \end{bmatrix}$	$\textcircled{D}$ [0.07, 1.00, 1.00] $\textcircled{D}_*$ [0.05, 0.71, 0.71]	$\Delta[\Xi_u]$ 0.2505 $\Delta[\textcircled{D}_*]^{\oplus}$ 0.0298
I_4	$\begin{bmatrix} 0.524 & 0.842 & -0.127 \\ -0.844 & 0.533 & 0.050 \\ 0.110 & 0.081 & 0.991 \end{bmatrix}$	$\textcircled{D}$ [0.14, 1.00, 0.99] $\textcircled{D}_*$ [0.10, 0.70, 0.70]	$\Delta[\Xi_u]$ 0.2816 $\Delta[\textcircled{D}_*]^{\oplus}$ 0.0615

(e) Hypothesis H_2 for $n_\theta = 10$.

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\textcircled{D}_*]^{\oplus}$
H_2	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 1.000 & 0.000 \\ 0.000 & 0.000 & 1.000 \end{bmatrix}$	$\textcircled{D}$ [0.00, 1.00, 1.00] $\textcircled{D}_*$ [0.00, 0.71, 0.71]	
I_0	$\begin{bmatrix} 0.443 & 0.896 & -0.026 \\ 0.896 & -0.443 & -0.001 \\ -0.013 & -0.023 & -1.000 \end{bmatrix}$	$\textcircled{D}$ [0.03, 1.00, 1.00] $\textcircled{D}_*$ [0.02, 0.71, 0.71]	$\Delta[\Xi_u]$ 0.3110 $\Delta[\textcircled{D}_*]^{\oplus}$ 0.0118
I_1	$\begin{bmatrix} -0.113 & -0.992 & 0.056 \\ -0.993 & 0.112 & -0.024 \\ 0.017 & -0.058 & -0.998 \end{bmatrix}$	$\textcircled{D}$ [0.06, 1.00, 1.00] $\textcircled{D}_*$ [0.04, 0.71, 0.71]	$\Delta[\Xi_u]$ 0.3394 $\Delta[\textcircled{D}_*]^{\oplus}$ 0.0273
I_2	$\begin{bmatrix} -0.311 & -0.950 & 0.014 \\ 0.949 & -0.310 & 0.051 \\ -0.044 & 0.029 & 0.999 \end{bmatrix}$	$\textcircled{D}$ [0.05, 1.00, 1.00] $\textcircled{D}_*$ [0.04, 0.71, 0.71]	$\Delta[\Xi_u]$ 0.3040 $\Delta[\textcircled{D}_*]^{\oplus}$ 0.0237
I_3	$\begin{bmatrix} -0.649 & -0.760 & 0.038 \\ 0.758 & -0.650 & -0.051 \\ 0.064 & -0.005 & 0.998 \end{bmatrix}$	$\textcircled{D}$ [0.06, 1.00, 1.00] $\textcircled{D}_*$ [0.05, 0.71, 0.71]	$\Delta[\Xi_u]$ 0.2468 $\Delta[\textcircled{D}_*]^{\oplus}$ 0.0287
I_4	$\begin{bmatrix} 0.731 & 0.650 & -0.209 \\ 0.645 & -0.758 & -0.101 \\ -0.224 & -0.061 & -0.973 \end{bmatrix}$	$\textcircled{D}$ [0.23, 1.00, 0.97] $\textcircled{D}_*$ [0.16, 0.71, 0.69]	$\Delta[\Xi_u]$ 0.2901 $\Delta[\textcircled{D}_*]^{\oplus}$ 0.1050

(f) Hypothesis H_2 for $n_\theta = 20$.

Table D.55: Noisy optimization parameterized with SO(n) on $\mathrm{H}_3^{\oplus}\mathrm{SE}(2)$. Sensor and actuator noise set to $\sigma_c, \sigma_p = 0.1$. True projection matrix and optimization results per iteration I_0 to I_4. Commutativity vector, unnormalized intrinsic conflict and commutativity delta.

Id	Hypothesis, $\Delta[\textcircled{D}_*]^{\oplus}$		
	H_0	H_1	H_2
I_0	0.0251	0.0176	0.0131
I_1	0.0143	0.0269	0.1072
I_2	0.0036	0.0126	0.0222
I_3	0.0297	0.1714	0.0298
I_4	0.0024	0.0083	0.0615
μ	0.0150	0.0474	0.0468
σ	0.0110	0.0623	0.0343
$\mu = 0.0364$, $\sigma = 0.0458$			

(a) Evaluation for $n_\theta = 10$.

Id	Hypothesis, $\Delta[\textcircled{D}_*]^{\oplus}$		
	H_0	H_1	H_2
I_0	0.0261	0.0139	0.0118
I_1	0.0136	0.0107	0.0273
I_2	0.0039	0.0034	0.0237
I_3	0.0257	0.1297	0.0287
I_4	0.0051	0.0091	0.1050
μ	0.0149	0.0334	0.0393
σ	0.0096	0.0483	0.0334
$\mu = 0.0292$, $\sigma = 0.0371$			

(b) Evaluation for $n_\theta = 20$.

Table D.56: Noisy optimization parameterized with SO(n) on $\mathrm{H}_3^{\oplus}\mathrm{SE}(2)$. Sensor and actuator noise set to $\sigma_c, \sigma_p = 0.10$. Commutativity delta $\Delta[\textcircled{D}_*]^{\oplus}$ per iteration I_0 to I_4. Mean and standard deviation of $\Delta[\textcircled{D}_*]^{\oplus}$ per hypothesis H_0 to H_2 and over all hypotheses.

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\textcircled{1}_*]^{\oplus}$
H_0	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 0.924 & -0.383 \\ 0.000 & 0.383 & 0.924 \end{bmatrix}$	$\textcircled{1}$ [0.38, 0.92, 1.00] $\textcircled{1}_*$ [0.27, 0.65, 0.71]	
I_0	$\begin{bmatrix} 0.720 & 0.674 & -0.166 \\ 0.605 & -0.492 & 0.626 \\ 0.341 & -0.551 & -0.762 \end{bmatrix}$	$\textcircled{1}$ [0.65, 0.83, 0.94] $\textcircled{1}_*$ [0.46, 0.59, 0.66]	$\Delta[\Xi_u]$ 0.6279 $\Delta[\textcircled{1}_*]^{\oplus}$ 0.1291
I_1	$\begin{bmatrix} -0.973 & 0.149 & 0.179 \\ 0.110 & 0.972 & -0.209 \\ -0.205 & -0.183 & -0.961 \end{bmatrix}$	$\textcircled{1}$ [0.28, 0.98, 0.98] $\textcircled{1}_*$ [0.19, 0.70, 0.69]	$\Delta[\Xi_u]$ 0.6297 $\Delta[\textcircled{1}_*]^{\oplus}$ 0.0561
I_2	$\begin{bmatrix} -0.817 & -0.570 & 0.088 \\ 0.495 & -0.615 & 0.614 \\ -0.296 & 0.545 & 0.784 \end{bmatrix}$	$\textcircled{1}$ [0.62, 0.84, 0.96] $\textcircled{1}_*$ [0.44, 0.59, 0.68]	$\Delta[\Xi_u]$ 0.6773 $\Delta[\textcircled{1}_*]^{\oplus}$ 0.1157
I_3	$\begin{bmatrix} -0.985 & -0.141 & -0.097 \\ 0.105 & -0.944 & 0.311 \\ -0.135 & 0.297 & 0.945 \end{bmatrix}$	$\textcircled{1}$ [0.33, 0.95, 0.99] $\textcircled{1}_*$ [0.23, 0.68, 0.70]	$\Delta[\Xi_u]$ 0.6055 $\Delta[\textcircled{1}_*]^{\oplus}$ 0.0294
I_4	$\begin{bmatrix} 0.046 & 0.931 & -0.363 \\ -0.994 & 0.007 & -0.108 \\ -0.097 & 0.366 & 0.925 \end{bmatrix}$	$\textcircled{1}$ [0.38, 0.93, 1.00] $\textcircled{1}_*$ [0.27, 0.66, 0.70]	$\Delta[\Xi_u]$ 0.6699 $\Delta[\textcircled{1}_*]^{\oplus}$ 0.0041

(a) Hypothesis H_0 for $n_\theta = 10$.

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\textcircled{1}_*]^{\oplus}$
H_0	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 0.924 & -0.383 \\ 0.000 & 0.383 & 0.924 \end{bmatrix}$	$\textcircled{1}$ [0.38, 0.92, 1.00] $\textcircled{1}_*$ [0.27, 0.65, 0.71]	
I_0	$\begin{bmatrix} 0.045 & -0.856 & 0.516 \\ 0.999 & 0.047 & -0.010 \\ -0.015 & 0.515 & 0.857 \end{bmatrix}$	$\textcircled{1}$ [0.52, 0.86, 1.00] $\textcircled{1}_*$ [0.36, 0.61, 0.71]	$\Delta[\Xi_u]$ 0.6272 $\Delta[\textcircled{1}_*]^{\oplus}$ 0.0671
I_1	$\begin{bmatrix} -0.985 & 0.099 & 0.140 \\ 0.073 & 0.981 & -0.179 \\ -0.155 & -0.167 & -0.974 \end{bmatrix}$	$\textcircled{1}$ [0.23, 0.99, 0.99] $\textcircled{1}_*$ [0.16, 0.70, 0.70]	$\Delta[\Xi_u]$ 0.6357 $\Delta[\textcircled{1}_*]^{\oplus}$ 0.0754
I_2	$\begin{bmatrix} -0.841 & -0.536 & 0.070 \\ 0.523 & -0.776 & 0.353 \\ -0.135 & 0.333 & 0.933 \end{bmatrix}$	$\textcircled{1}$ [0.36, 0.94, 0.99] $\textcircled{1}_*$ [0.25, 0.67, 0.70]	$\Delta[\Xi_u]$ 0.6740 $\Delta[\textcircled{1}_*]^{\oplus}$ 0.0141
I_3	$\begin{bmatrix} -0.983 & -0.185 & -0.012 \\ 0.172 & -0.936 & 0.308 \\ -0.068 & 0.300 & 0.951 \end{bmatrix}$	$\textcircled{1}$ [0.31, 0.95, 1.00] $\textcircled{1}_*$ [0.22, 0.67, 0.71]	$\Delta[\Xi_u]$ 0.6032 $\Delta[\textcircled{1}_*]^{\oplus}$ 0.0363
I_4	$\begin{bmatrix} 0.152 & 0.832 & -0.534 \\ 0.980 & -0.058 & 0.190 \\ 0.127 & -0.552 & -0.824 \end{bmatrix}$	$\textcircled{1}$ [0.57, 0.83, 0.99] $\textcircled{1}_*$ [0.40, 0.59, 0.70]	$\Delta[\Xi_u]$ 0.6634 $\Delta[\textcircled{1}_*]^{\oplus}$ 0.0924

(b) Hypothesis H_0 for $n_\theta = 20$.

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\textcircled{1}_*]^{\oplus}$
H_1	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 0.707 & -0.707 \\ 0.000 & 0.707 & 0.707 \end{bmatrix}$	$\textcircled{1}$ [0.71, 0.71, 1.00] $\textcircled{1}_*$ [0.50, 0.50, 0.71]	
I_0	$\begin{bmatrix} -0.871 & -0.220 & 0.439 \\ -0.488 & 0.482 & -0.728 \\ -0.052 & -0.848 & -0.527 \end{bmatrix}$	$\textcircled{1}$ [0.85, 0.53, 1.00] $\textcircled{1}_*$ [0.60, 0.37, 0.71]	$\Delta[\Xi_u]$ 0.6287 $\Delta[\textcircled{1}_*]^{\oplus}$ 0.1025
I_1	$\begin{bmatrix} -0.619 & 0.671 & -0.408 \\ -0.784 & -0.506 & 0.359 \\ 0.034 & 0.543 & 0.839 \end{bmatrix}$	$\textcircled{1}$ [0.54, 0.84, 1.00] $\textcircled{1}_*$ [0.38, 0.59, 0.71]	$\Delta[\Xi_u]$ 0.6324 $\Delta[\textcircled{1}_*]^{\oplus}$ 0.0949
I_2	$\begin{bmatrix} -0.997 & 0.078 & 0.010 \\ -0.051 & -0.741 & 0.669 \\ 0.059 & 0.667 & 0.743 \end{bmatrix}$	$\textcircled{1}$ [0.67, 0.75, 1.00] $\textcircled{1}_*$ [0.47, 0.53, 0.71]	$\Delta[\Xi_u]$ 0.6278 $\Delta[\textcircled{1}_*]^{\oplus}$ 0.0241
I_3	$\begin{bmatrix} -0.711 & -0.512 & 0.482 \\ 0.673 & -0.695 & 0.253 \\ 0.205 & 0.504 & 0.839 \end{bmatrix}$	$\textcircled{1}$ [0.54, 0.86, 0.98] $\textcircled{1}_*$ [0.38, 0.61, 0.69]	$\Delta[\Xi_u]$ 0.5967 $\Delta[\textcircled{1}_*]^{\oplus}$ 0.1022
I_4	$\begin{bmatrix} -0.577 & 0.410 & -0.706 \\ 0.807 & 0.415 & -0.419 \\ 0.122 & -0.812 & -0.571 \end{bmatrix}$	$\textcircled{1}$ [0.82, 0.58, 0.99] $\textcircled{1}_*$ [0.58, 0.41, 0.70]	$\Delta[\Xi_u]$ 0.6329 $\Delta[\textcircled{1}_*]^{\oplus}$ 0.0758

(c) Hypothesis H_1 for $n_\theta = 10$.

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\textcircled{1}_*]^{\oplus}$
H_1	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 0.707 & -0.707 \\ 0.000 & 0.707 & 0.707 \end{bmatrix}$	$\textcircled{1}$ [0.71, 0.71, 1.00] $\textcircled{1}_*$ [0.50, 0.50, 0.71]	
I_0	$\begin{bmatrix} -0.970 & -0.195 & 0.144 \\ -0.240 & 0.695 & -0.677 \\ 0.032 & -0.692 & -0.722 \end{bmatrix}$	$\textcircled{1}$ [0.69, 0.72, 1.00] $\textcircled{1}_*$ [0.49, 0.51, 0.71]	$\Delta[\Xi_u]$ 0.6254 $\Delta[\textcircled{1}_*]^{\oplus}$ 0.0095
I_1	$\begin{bmatrix} 0.915 & -0.401 & 0.030 \\ -0.314 & -0.666 & 0.676 \\ -0.252 & -0.629 & -0.736 \end{bmatrix}$	$\textcircled{1}$ [0.68, 0.78, 0.97] $\textcircled{1}_*$ [0.48, 0.55, 0.68]	$\Delta[\Xi_u]$ 0.6562 $\Delta[\textcircled{1}_*]^{\oplus}$ 0.0375
I_2	$\begin{bmatrix} -0.974 & -0.136 & 0.182 \\ 0.222 & -0.745 & 0.630 \\ 0.050 & 0.653 & 0.755 \end{bmatrix}$	$\textcircled{1}$ [0.66, 0.76, 1.00] $\textcircled{1}_*$ [0.46, 0.54, 0.71]	$\Delta[\Xi_u]$ 0.6304 $\Delta[\textcircled{1}_*]^{\oplus}$ 0.0323
I_3	$\begin{bmatrix} 0.018 & -0.849 & 0.528 \\ -0.979 & 0.092 & 0.181 \\ -0.202 & -0.520 & -0.830 \end{bmatrix}$	$\textcircled{1}$ [0.56, 0.85, 0.98] $\textcircled{1}_*$ [0.39, 0.60, 0.69]	$\Delta[\Xi_u]$ 0.5983 $\Delta[\textcircled{1}_*]^{\oplus}$ 0.0947
I_4	$\begin{bmatrix} -0.003 & 0.644 & -0.765 \\ 0.931 & -0.278 & -0.238 \\ -0.366 & -0.713 & -0.598 \end{bmatrix}$	$\textcircled{1}$ [0.80, 0.70, 0.93] $\textcircled{1}_*$ [0.57, 0.50, 0.66]	$\Delta[\Xi_u]$ 0.6301 $\Delta[\textcircled{1}_*]^{\oplus}$ 0.0527

(d) Hypothesis H_1 for $n_\theta = 20$.

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\textcircled{1}_*]^{\oplus}$
H_2	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 1.000 & 0.000 \\ 0.000 & 0.000 & 1.000 \end{bmatrix}$	$\textcircled{1}$ [0.00, 1.00, 1.00] $\textcircled{1}_*$ [0.00, 0.71, 0.71]	
I_0	$\begin{bmatrix} 0.591 & 0.804 & -0.066 \\ 0.802 & -0.577 & 0.151 \\ 0.083 & -0.142 & -0.986 \end{bmatrix}$	$\textcircled{1}$ [0.17, 0.99, 1.00] $\textcircled{1}_*$ [0.12, 0.70, 0.70]	$\Delta[\Xi_u]$ 0.6855 $\Delta[\textcircled{1}_*]^{\oplus}$ 0.0745
I_1	$\begin{bmatrix} 0.112 & -0.938 & 0.328 \\ -0.992 & -0.086 & 0.092 \\ -0.059 & -0.336 & -0.940 \end{bmatrix}$	$\textcircled{1}$ [0.34, 0.94, 1.00] $\textcircled{1}_*$ [0.24, 0.67, 0.71]	$\Delta[\Xi_u]$ 0.6939 $\Delta[\textcircled{1}_*]^{\oplus}$ 0.1562
I_2	$\begin{bmatrix} 0.141 & 0.984 & 0.108 \\ 0.621 & -0.003 & -0.784 \\ -0.771 & 0.177 & -0.612 \end{bmatrix}$	$\textcircled{1}$ [0.79, 0.98, 0.64] $\textcircled{1}_*$ [0.56, 0.70, 0.45]	$\Delta[\Xi_u]$ 0.6451 $\Delta[\textcircled{1}_*]^{\oplus}$ 0.3985
I_3	$\begin{bmatrix} -0.979 & -0.202 & -0.001 \\ 0.202 & -0.978 & -0.046 \\ 0.009 & -0.046 & 0.999 \end{bmatrix}$	$\textcircled{1}$ [0.05, 1.00, 1.00] $\textcircled{1}_*$ [0.03, 0.71, 0.71]	$\Delta[\Xi_u]$ 0.5982 $\Delta[\textcircled{1}_*]^{\oplus}$ 0.0209
I_4	$\begin{bmatrix} 0.896 & -0.438 & -0.080 \\ 0.424 & 0.894 & -0.144 \\ 0.135 & 0.095 & 0.986 \end{bmatrix}$	$\textcircled{1}$ [0.16, 1.00, 0.99] $\textcircled{1}_*$ [0.12, 0.70, 0.70]	$\Delta[\Xi_u]$ 0.6444 $\Delta[\textcircled{1}_*]^{\oplus}$ 0.0744

(e) Hypothesis H_2 for $n_\theta = 10$.

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\textcircled{1}_*]^{\oplus}$
H_2	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 1.000 & 0.000 \\ 0.000 & 0.000 & 1.000 \end{bmatrix}$	$\textcircled{1}$ [0.00, 1.00, 1.00] $\textcircled{1}_*$ [0.00, 0.71, 0.71]	
I_0	$\begin{bmatrix} 0.588 & 0.809 & -0.009 \\ -0.790 & 0.572 & -0.222 \\ -0.174 & 0.138 & 0.975 \end{bmatrix}$	$\textcircled{1}$ [0.22, 0.99, 0.98] $\textcircled{1}_*$ [0.16, 0.70, 0.70]	$\Delta[\Xi_u]$ 0.6822 $\Delta[\textcircled{1}_*]^{\oplus}$ 0.1004
I_1	$\begin{bmatrix} -0.373 & -0.925 & -0.073 \\ -0.925 & 0.364 & 0.111 \\ -0.076 & 0.109 & -0.991 \end{bmatrix}$	$\textcircled{1}$ [0.13, 0.99, 1.00] $\textcircled{1}_*$ [0.09, 0.70, 0.71]	$\Delta[\Xi_u]$ 0.6876 $\Delta[\textcircled{1}_*]^{\oplus}$ 0.0599
I_2	$\begin{bmatrix} -0.985 & 0.041 & 0.165 \\ 0.044 & 0.999 & 0.018 \\ -0.164 & 0.025 & -0.986 \end{bmatrix}$	$\textcircled{1}$ [0.17, 1.00, 0.99] $\textcircled{1}_*$ [0.12, 0.71, 0.70]	$\Delta[\Xi_u]$ 0.6122 $\Delta[\textcircled{1}_*]^{\oplus}$ 0.0750
I_3	$\begin{bmatrix} -0.915 & -0.404 & -0.017 \\ 0.404 & -0.914 & -0.045 \\ 0.003 & -0.048 & 0.999 \end{bmatrix}$	$\textcircled{1}$ [0.05, 1.00, 1.00] $\textcircled{1}_*$ [0.03, 0.71, 0.71]	$\Delta[\Xi_u]$ 0.6012 $\Delta[\textcircled{1}_*]^{\oplus}$ 0.0217
I_4	$\begin{bmatrix} -0.779 & -0.624 & 0.058 \\ -0.627 & 0.776 & -0.063 \\ -0.006 & -0.086 & -0.996 \end{bmatrix}$	$\textcircled{1}$ [0.09, 1.00, 1.00] $\textcircled{1}_*$ [0.06, 0.70, 0.71]	$\Delta[\Xi_u]$ 0.6408 $\Delta[\textcircled{1}_*]^{\oplus}$ 0.0389

(f) Hypothesis H_2 for $n_\theta = 20$.

Table D.57: Noisy optimization parameterized with SO(n) on $\mathrm{H}_3^{\oplus}\mathrm{SE}(2)$. Sensor and actuator noise set to $\sigma_c, \sigma_p = 0.25$. True projection matrix and optimization results per iteration I_0 to I_4. Commutativity vector, unnormalized intrinsic conflict and commutativity delta.

Id	Hypothesis, $\Delta[\textcircled{1}_*]^{\oplus}$		
	H_0	H_1	H_2
I_0	0.1291	0.1025	0.0745
I_1	0.0561	0.0949	0.1562
I_2	0.1157	0.0241	0.3985
I_3	0.0294	0.1022	0.0209
I_4	0.0041	0.0758	0.0744
μ	0.0669	0.0799	0.1449
σ	0.0484	0.0295	0.1340
$\mu = 0.0972$, $\sigma = 0.0938$			

(a) Evaluation for $n_\theta = 10$.

Id	Hypothesis, $\Delta[\textcircled{1}_*]^{\oplus}$		
	H_0	H_1	H_2
I_0	0.0671	0.0095	0.1004
I_1	0.0754	0.0375	0.0599
I_2	0.0141	0.0323	0.0750
I_3	0.0363	0.0947	0.0217
I_4	0.0924	0.0527	0.0389
μ	0.0571	0.0453	0.0592
σ	0.0282	0.0283	0.0274
$\mu = 0.0539$, $\sigma = 0.0296$			

(b) Evaluation for $n_\theta = 20$.

Table D.58: Noisy optimization parameterized with SO(n) on $\mathrm{H}_3^{\oplus}\mathrm{SE}(2)$. Sensor and actuator noise set to $\sigma_c, \sigma_p = 0.25$. Commutativity delta $\Delta[\textcircled{1}_*]^{\oplus}$ per iteration I_0 to I_4. Mean and standard deviation of $\Delta[\textcircled{1}_*]^{\oplus}$ per hypothesis H_0 to H_2 and over all hypotheses.

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\textcircled{D}_*]^{\oplus}$
H_0	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 0.924 & -0.383 \\ 0.000 & 0.383 & 0.924 \end{bmatrix}$	$\textcircled{D}$ [0.38, 0.92, 1.00] $\textcircled{D}_*$ [0.27, 0.65, 0.71]	
I_0	$\begin{bmatrix} 0.270 & 0.933 & 0.238 \\ -0.845 & 0.111 & 0.523 \\ 0.462 & -0.342 & 0.818 \end{bmatrix}$	$\textcircled{D}$ [0.57, 0.94, 0.89] $\textcircled{D}_*$ [0.41, 0.66, 0.63]	$\Delta[\Xi_u]$ 1.7207 $\Delta[\textcircled{D}_*]^{\oplus}$ 0.1006
I_1	$\begin{bmatrix} 0.730 & -0.669 & 0.140 \\ 0.634 & 0.585 & -0.506 \\ 0.256 & 0.458 & 0.851 \end{bmatrix}$	$\textcircled{D}$ [0.52, 0.89, 0.97] $\textcircled{D}_*$ [0.37, 0.63, 0.68]	$\Delta[\Xi_u]$ 1.7142 $\Delta[\textcircled{D}_*]^{\oplus}$ 0.0677
I_2	$\begin{bmatrix} 0.365 & 0.749 & 0.553 \\ -0.612 & 0.641 & -0.464 \\ -0.702 & -0.169 & 0.692 \end{bmatrix}$	$\textcircled{D}$ [0.72, 0.99, 0.71] $\textcircled{D}_*$ [0.51, 0.70, 0.50]	$\Delta[\Xi_u]$ 1.7509 $\Delta[\textcircled{D}_*]^{\oplus}$ 0.2029
I_3	$\begin{bmatrix} -0.729 & -0.631 & -0.264 \\ -0.684 & 0.665 & 0.300 \\ -0.014 & 0.400 & -0.917 \end{bmatrix}$	$\textcircled{D}$ [0.40, 0.92, 1.00] $\textcircled{D}_*$ [0.28, 0.65, 0.71]	$\Delta[\Xi_u]$ 1.7535 $\Delta[\textcircled{D}_*]^{\oplus}$ 0.0085
I_4	$\begin{bmatrix} 0.709 & 0.286 & -0.645 \\ -0.053 & 0.933 & 0.355 \\ 0.704 & -0.217 & 0.677 \end{bmatrix}$	$\textcircled{D}$ [0.74, 0.98, 0.71] $\textcircled{D}_*$ [0.52, 0.69, 0.50]	$\Delta[\Xi_u]$ 1.7516 $\Delta[\textcircled{D}_*]^{\oplus}$ 0.2080

(a) Hypothesis H_0 for $n_\theta = 10$.

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\textcircled{D}_*]^{\oplus}$
H_0	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 0.924 & -0.383 \\ 0.000 & 0.383 & 0.924 \end{bmatrix}$	$\textcircled{D}$ [0.38, 0.92, 1.00] $\textcircled{D}_*$ [0.27, 0.65, 0.71]	
I_0	$\begin{bmatrix} 0.364 & -0.593 & -0.719 \\ -0.931 & -0.247 & -0.267 \\ -0.019 & 0.767 & -0.642 \end{bmatrix}$	$\textcircled{D}$ [0.77, 0.64, 1.00] $\textcircled{D}_*$ [0.54, 0.45, 0.71]	$\Delta[\Xi_u]$ 1.7202 $\Delta[\textcircled{D}_*]^{\oplus}$ 0.2154
I_1	$\begin{bmatrix} 0.757 & -0.574 & 0.313 \\ -0.307 & 0.111 & 0.945 \\ -0.577 & -0.811 & -0.092 \end{bmatrix}$	$\textcircled{D}$ [1.00, 0.58, 0.82] $\textcircled{D}_*$ [0.70, 0.41, 0.58]	$\Delta[\Xi_u]$ 1.6945 $\Delta[\textcircled{D}_*]^{\oplus}$ 0.3298
I_2	$\begin{bmatrix} -0.551 & 0.834 & -0.031 \\ -0.726 & -0.497 & -0.475 \\ -0.411 & -0.239 & 0.880 \end{bmatrix}$	$\textcircled{D}$ [0.48, 0.97, 0.91] $\textcircled{D}_*$ [0.34, 0.69, 0.64]	$\Delta[\Xi_u]$ 1.7331 $\Delta[\textcircled{D}_*]^{\oplus}$ 0.0616
I_3	$\begin{bmatrix} -0.759 & -0.606 & -0.239 \\ -0.651 & 0.698 & 0.298 \\ -0.014 & 0.382 & -0.924 \end{bmatrix}$	$\textcircled{D}$ [0.38, 0.92, 1.00] $\textcircled{D}_*$ [0.27, 0.65, 0.71]	$\Delta[\Xi_u]$ 1.7564 $\Delta[\textcircled{D}_*]^{\oplus}$ 0.0004
I_4	$\begin{bmatrix} -0.732 & -0.411 & 0.543 \\ 0.023 & 0.782 & 0.623 \\ -0.680 & 0.469 & -0.563 \end{bmatrix}$	$\textcircled{D}$ [0.83, 0.88, 0.73] $\textcircled{D}_*$ [0.58, 0.62, 0.52]	$\Delta[\Xi_u]$ 1.7417 $\Delta[\textcircled{D}_*]^{\oplus}$ 0.2352

(b) Hypothesis H_0 for $n_\theta = 20$.

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\textcircled{D}_*]^{\oplus}$
H_1	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 0.707 & -0.707 \\ 0.000 & 0.707 & 0.707 \end{bmatrix}$	$\textcircled{D}$ [0.71, 0.71, 1.00] $\textcircled{D}_*$ [0.50, 0.50, 0.71]	
I_0	$\begin{bmatrix} -0.140 & 0.837 & 0.529 \\ -0.842 & 0.181 & -0.508 \\ -0.521 & -0.517 & 0.680 \end{bmatrix}$	$\textcircled{D}$ [0.73, 0.86, 0.85] $\textcircled{D}_*$ [0.52, 0.61, 0.60]	$\Delta[\Xi_u]$ 1.7743 $\Delta[\textcircled{D}_*]^{\oplus}$ 0.0949
I_1	$\begin{bmatrix} 0.877 & 0.309 & -0.368 \\ -0.476 & 0.652 & -0.590 \\ 0.058 & 0.692 & 0.719 \end{bmatrix}$	$\textcircled{D}$ [0.69, 0.72, 1.00] $\textcircled{D}_*$ [0.49, 0.51, 0.71]	$\Delta[\Xi_u]$ 1.6967 $\Delta[\textcircled{D}_*]^{\oplus}$ 0.0085
I_2	$\begin{bmatrix} 0.412 & -0.826 & -0.385 \\ -0.187 & -0.490 & 0.851 \\ -0.892 & -0.279 & -0.356 \end{bmatrix}$	$\textcircled{D}$ [0.93, 0.96, 0.45] $\textcircled{D}_*$ [0.66, 0.68, 0.32]	$\Delta[\Xi_u]$ 1.7113 $\Delta[\textcircled{D}_*]^{\oplus}$ 0.2928
I_3	$\begin{bmatrix} -0.582 & -0.801 & -0.139 \\ 0.679 & -0.573 & 0.459 \\ -0.447 & 0.173 & 0.877 \end{bmatrix}$	$\textcircled{D}$ [0.48, 0.98, 0.89] $\textcircled{D}_*$ [0.34, 0.70, 0.63]	$\Delta[\Xi_u]$ 1.7125 $\Delta[\textcircled{D}_*]^{\oplus}$ 0.1689
I_4	$\begin{bmatrix} -0.156 & 0.868 & -0.471 \\ 0.963 & 0.027 & -0.269 \\ -0.221 & -0.496 & -0.840 \end{bmatrix}$	$\textcircled{D}$ [0.54, 0.87, 0.98] $\textcircled{D}_*$ [0.38, 0.61, 0.69]	$\Delta[\Xi_u]$ 1.7065 $\Delta[\textcircled{D}_*]^{\oplus}$ 0.1045

(c) Hypothesis H_1 for $n_\theta = 10$.

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\textcircled{D}_*]^{\oplus}$
H_1	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 0.707 & -0.707 \\ 0.000 & 0.707 & 0.707 \end{bmatrix}$	$\textcircled{D}$ [0.71, 0.71, 1.00] $\textcircled{D}_*$ [0.50, 0.50, 0.71]	
I_0	$\begin{bmatrix} 0.598 & 0.402 & 0.693 \\ 0.649 & -0.751 & -0.124 \\ 0.470 & 0.524 & -0.710 \end{bmatrix}$	$\textcircled{D}$ [0.70, 0.85, 0.88] $\textcircled{D}_*$ [0.50, 0.60, 0.62]	$\Delta[\Xi_u]$ 1.7434 $\Delta[\textcircled{D}_*]^{\oplus}$ 0.0840
I_1	$\begin{bmatrix} 0.714 & -0.635 & 0.297 \\ -0.700 & -0.635 & 0.326 \\ -0.019 & -0.441 & -0.898 \end{bmatrix}$	$\textcircled{D}$ [0.44, 0.90, 1.00] $\textcircled{D}_*$ [0.31, 0.63, 0.71]	$\Delta[\Xi_u]$ 1.7134 $\Delta[\textcircled{D}_*]^{\oplus}$ 0.1477
I_2	$\begin{bmatrix} 0.333 & -0.940 & -0.079 \\ -0.309 & -0.188 & 0.932 \\ -0.891 & -0.286 & -0.353 \end{bmatrix}$	$\textcircled{D}$ [0.94, 0.96, 0.45] $\textcircled{D}_*$ [0.66, 0.68, 0.32]	$\Delta[\Xi_u]$ 1.6791 $\Delta[\textcircled{D}_*]^{\oplus}$ 0.2917
I_3	$\begin{bmatrix} -0.503 & 0.761 & -0.409 \\ 0.740 & 0.624 & 0.249 \\ 0.445 & -0.178 & -0.878 \end{bmatrix}$	$\textcircled{D}$ [0.48, 0.98, 0.90] $\textcircled{D}_*$ [0.34, 0.70, 0.63]	$\Delta[\Xi_u]$ 1.7014 $\Delta[\textcircled{D}_*]^{\oplus}$ 0.1686
I_4	$\begin{bmatrix} 0.958 & 0.104 & 0.269 \\ 0.132 & 0.672 & -0.728 \\ -0.256 & 0.733 & 0.630 \end{bmatrix}$	$\textcircled{D}$ [0.78, 0.68, 0.97] $\textcircled{D}_*$ [0.55, 0.48, 0.68]	$\Delta[\Xi_u]$ 1.6927 $\Delta[\textcircled{D}_*]^{\oplus}$ 0.0367

(d) Hypothesis H_1 for $n_\theta = 20$.

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\textcircled{D}_*]^{\oplus}$
H_2	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 1.000 & 0.000 \\ 0.000 & 0.000 & 1.000 \end{bmatrix}$	$\textcircled{D}$ [0.00, 1.00, 1.00] $\textcircled{D}_*$ [0.00, 0.71, 0.71]	
I_0	$\begin{bmatrix} 0.220 & 0.805 & 0.551 \\ 0.274 & 0.491 & -0.827 \\ -0.936 & 0.333 & -0.113 \end{bmatrix}$	$\textcircled{D}$ [0.99, 0.94, 0.35] $\textcircled{D}_*$ [0.70, 0.67, 0.25]	$\Delta[\Xi_u]$ 1.8676 $\Delta[\textcircled{D}_*]^{\oplus}$ 0.5519
I_1	$\begin{bmatrix} 0.743 & -0.620 & -0.253 \\ 0.304 & -0.024 & 0.952 \\ -0.596 & -0.785 & 0.170 \end{bmatrix}$	$\textcircled{D}$ [0.99, 0.62, 0.80] $\textcircled{D}_*$ [0.70, 0.44, 0.57]	$\Delta[\Xi_u]$ 1.7419 $\Delta[\textcircled{D}_*]^{\oplus}$ 0.4961
I_2	$\begin{bmatrix} 0.961 & 0.264 & -0.079 \\ 0.207 & -0.882 & -0.424 \\ -0.182 & 0.391 & -0.902 \end{bmatrix}$	$\textcircled{D}$ [0.43, 0.92, 0.98] $\textcircled{D}_*$ [0.31, 0.65, 0.70]	$\Delta[\Xi_u]$ 1.7918 $\Delta[\textcircled{D}_*]^{\oplus}$ 0.1985
I_3	$\begin{bmatrix} 0.858 & -0.512 & -0.047 \\ 0.500 & 0.812 & 0.300 \\ -0.116 & -0.281 & 0.953 \end{bmatrix}$	$\textcircled{D}$ [0.30, 0.96, 0.99] $\textcircled{D}_*$ [0.21, 0.68, 0.70]	$\Delta[\Xi_u]$ 1.7609 $\Delta[\textcircled{D}_*]^{\oplus}$ 0.1384
I_4	$\begin{bmatrix} -0.838 & 0.466 & -0.282 \\ -0.069 & 0.423 & 0.903 \\ 0.541 & 0.777 & -0.323 \end{bmatrix}$	$\textcircled{D}$ [0.95, 0.63, 0.84] $\textcircled{D}_*$ [0.67, 0.45, 0.59]	$\Delta[\Xi_u]$ 1.7579 $\Delta[\textcircled{D}_*]^{\oplus}$ 0.4740

(e) Hypothesis H_2 for $n_\theta = 10$.

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\textcircled{D}_*]^{\oplus}$
H_2	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 1.000 & 0.000 \\ 0.000 & 0.000 & 1.000 \end{bmatrix}$	$\textcircled{D}$ [0.00, 1.00, 1.00] $\textcircled{D}_*$ [0.00, 0.71, 0.71]	
I_0	$\begin{bmatrix} 0.088 & -0.671 & -0.736 \\ -0.989 & -0.144 & 0.013 \\ -0.115 & 0.727 & -0.677 \end{bmatrix}$	$\textcircled{D}$ [0.74, 0.69, 0.99] $\textcircled{D}_*$ [0.52, 0.49, 0.70]	$\Delta[\Xi_u]$ 1.8481 $\Delta[\textcircled{D}_*]^{\oplus}$ 0.3652
I_1	$\begin{bmatrix} 0.462 & -0.503 & -0.730 \\ -0.668 & 0.345 & -0.660 \\ 0.584 & 0.793 & -0.177 \end{bmatrix}$	$\textcircled{D}$ [0.98, 0.61, 0.81] $\textcircled{D}_*$ [0.70, 0.43, 0.57]	$\Delta[\Xi_u]$ 1.7067 $\Delta[\textcircled{D}_*]^{\oplus}$ 0.4966
I_2	$\begin{bmatrix} -0.455 & -0.842 & -0.290 \\ 0.862 & -0.498 & 0.095 \\ -0.224 & -0.207 & 0.952 \end{bmatrix}$	$\textcircled{D}$ [0.30, 0.98, 0.97] $\textcircled{D}_*$ [0.22, 0.69, 0.69]	$\Delta[\Xi_u]$ 1.7727 $\Delta[\textcircled{D}_*]^{\oplus}$ 0.1384
I_3	$\begin{bmatrix} 0.925 & -0.222 & 0.308 \\ 0.289 & -0.114 & -0.951 \\ 0.246 & 0.968 & -0.041 \end{bmatrix}$	$\textcircled{D}$ [1.00, 0.25, 0.97] $\textcircled{D}_*$ [0.71, 0.18, 0.69]	$\Delta[\Xi_u]$ 1.7929 $\Delta[\textcircled{D}_*]^{\oplus}$ 0.5828
I_4	$\begin{bmatrix} -0.544 & 0.821 & -0.176 \\ 0.429 & 0.452 & 0.782 \\ 0.721 & 0.350 & -0.598 \end{bmatrix}$	$\textcircled{D}$ [0.80, 0.94, 0.69] $\textcircled{D}_*$ [0.57, 0.66, 0.49]	$\Delta[\Xi_u]$ 1.7754 $\Delta[\textcircled{D}_*]^{\oplus}$ 0.3937

(f) Hypothesis H_2 for $n_\theta = 20$.

Table D.59: Noisy optimization parameterized with SO(n) on $\mathrm{H}_3^{\oplus}\mathrm{SE}(2)$. Sensor and actuator noise set to $\sigma_c, \sigma_p = 0.50$. True projection matrix and optimization results per iteration I_0 to I_4. Commutativity vector, unnormalized intrinsic conflict and commutativity delta.

Id	Hypothesis, $\Delta[\textcircled{D}_*]^{\oplus}$		
	H_0	H_1	H_2
I_0	0.1006	0.0949	0.5519
I_1	0.0677	0.0085	0.4961
I_2	0.2029	0.2928	0.1985
I_3	0.0085	0.1689	0.1384
I_4	0.2080	0.1045	0.4740
μ	0.1175	0.1339	0.3718
σ	0.0776	0.0944	0.1690
$\mu = 0.2077$, $\sigma = 0.1732$			

(a) Evaluation for $n_\theta = 10$.

Id	Hypothesis, $\Delta[\textcircled{D}_*]^{\oplus}$		
	H_0	H_1	H_2
I_0	0.2154	0.0840	0.3652
I_1	0.3298	0.1477	0.4966
I_2	0.0616	0.2917	0.1384
I_3	0.0004	0.1686	0.5828
I_4	0.2352	0.0367	0.3937
μ	0.1685	0.1457	0.3953
σ	0.1203	0.0866	0.1499
$\mu = 0.2365$, $\sigma = 0.1717$			

(b) Evaluation for $n_\theta = 20$.

Table D.60: Noisy optimization parameterized with SO(n) on $\mathrm{H}_3^{\oplus}\mathrm{SE}(2)$. Sensor and actuator noise set to $\sigma_c, \sigma_p = 0.50$. Commutativity delta $\Delta[\textcircled{D}_*]^{\oplus}$ per iteration I_0 to I_4. Mean and standard deviation of $\Delta[\textcircled{D}_*]^{\oplus}$ per hypothesis H_0 to H_2 and over all hypotheses.

D.11 Scenario VIb : Nonholonomic Optimization on SE(2)

D.11.1 Non-Noisy Optimization parameterized with SO(n) on $\mathrm{H}_3^{\ominus}\mathrm{SE}(2)$

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\mathbb{D}_*]^{\ominus}$
H_0	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 0.924 & -0.383 \\ 0.000 & 0.383 & 0.924 \end{bmatrix}$	$\mathbb{D}$ [0.38] -	
I_0	$\begin{bmatrix} 0.587 & 0.751 & -0.302 \\ 0.810 & -0.545 & 0.219 \\ -0.000 & -0.373 & -0.928 \end{bmatrix}$	$\mathbb{D}$ [0.37] -	$\Delta[\Xi_u]$ 0.1153 $\Delta[\mathbb{D}_*]^{\ominus}$ 0.0092
I_1	$\begin{bmatrix} -0.991 & -0.004 & 0.134 \\ -0.040 & 0.963 & -0.265 \\ -0.128 & -0.268 & -0.955 \end{bmatrix}$	$\mathbb{D}$ [0.30] -	$\Delta[\Xi_u]$ 0.1676 $\Delta[\mathbb{D}_*]^{\ominus}$ 0.0857
I_2	$\begin{bmatrix} 0.700 & 0.660 & -0.272 \\ -0.714 & 0.652 & -0.257 \\ 0.007 & 0.374 & 0.927 \end{bmatrix}$	$\mathbb{D}$ [0.37] -	$\Delta[\Xi_u]$ 0.0897 $\Delta[\mathbb{D}_*]^{\ominus}$ 0.0086
I_3	$\begin{bmatrix} -0.795 & 0.563 & -0.226 \\ 0.607 & 0.734 & -0.305 \\ -0.006 & -0.380 & -0.925 \end{bmatrix}$	$\mathbb{D}$ [0.38] -	$\Delta[\Xi_u]$ 0.0893 $\Delta[\mathbb{D}_*]^{\ominus}$ 0.0025
I_4	$\begin{bmatrix} -0.043 & 0.920 & -0.389 \\ 0.999 & 0.038 & -0.020 \\ -0.004 & -0.390 & -0.921 \end{bmatrix}$	$\mathbb{D}$ [0.39] -	$\Delta[\Xi_u]$ 0.0910 $\Delta[\mathbb{D}_*]^{\ominus}$ 0.0069

(a) Hypothesis H_0 for $n_\theta = 10$.

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\mathbb{D}_*]^{\ominus}$
H_0	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 0.924 & -0.383 \\ 0.000 & 0.383 & 0.924 \end{bmatrix}$	$\mathbb{D}$ [0.38] -	
I_0	$\begin{bmatrix} 0.936 & 0.325 & -0.134 \\ 0.352 & -0.866 & 0.355 \\ -0.001 & -0.379 & -0.925 \end{bmatrix}$	$\mathbb{D}$ [0.38] -	$\Delta[\Xi_u]$ 0.1134 $\Delta[\mathbb{D}_*]^{\ominus}$ 0.0033
I_1	$\begin{bmatrix} -0.346 & -0.867 & 0.360 \\ -0.938 & 0.316 & -0.142 \\ 0.009 & -0.386 & -0.922 \end{bmatrix}$	$\mathbb{D}$ [0.39] -	$\Delta[\Xi_u]$ 0.1131 $\Delta[\mathbb{D}_*]^{\ominus}$ 0.0037
I_2	$\begin{bmatrix} -0.928 & -0.344 & 0.142 \\ -0.372 & 0.859 & -0.353 \\ -0.001 & -0.380 & -0.925 \end{bmatrix}$	$\mathbb{D}$ [0.38] -	$\Delta[\Xi_u]$ 0.0821 $\Delta[\mathbb{D}_*]^{\ominus}$ 0.0027
I_3	$\begin{bmatrix} -0.622 & 0.718 & -0.311 \\ 0.783 & 0.574 & -0.241 \\ 0.005 & -0.393 & -0.919 \end{bmatrix}$	$\mathbb{D}$ [0.39] -	$\Delta[\Xi_u]$ 0.0898 $\Delta[\mathbb{D}_*]^{\ominus}$ 0.0108
I_4	$\begin{bmatrix} -0.307 & 0.881 & -0.361 \\ 0.952 & 0.285 & -0.114 \\ 0.002 & -0.378 & -0.926 \end{bmatrix}$	$\mathbb{D}$ [0.38] -	$\Delta[\Xi_u]$ 0.0875 $\Delta[\mathbb{D}_*]^{\ominus}$ 0.0045

(b) Hypothesis H_0 for $n_\theta = 20$.

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\mathbb{D}_*]^{\ominus}$
H_1	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 0.707 & -0.707 \\ 0.000 & 0.707 & 0.707 \end{bmatrix}$	$\mathbb{D}$ [0.71] -	
I_0	$\begin{bmatrix} -1.000 & -0.021 & 0.020 \\ -0.029 & 0.707 & -0.707 \\ 0.001 & -0.707 & -0.707 \end{bmatrix}$	$\mathbb{D}$ [0.71] -	$\Delta[\Xi_u]$ 0.0982 $\Delta[\mathbb{D}_*]^{\ominus}$ 0.0002
I_1	$\begin{bmatrix} -0.511 & -0.610 & 0.606 \\ 0.860 & -0.362 & 0.361 \\ -0.001 & 0.705 & 0.709 \end{bmatrix}$	$\mathbb{D}$ [0.71] -	$\Delta[\Xi_u]$ 0.1016 $\Delta[\mathbb{D}_*]^{\ominus}$ 0.0019
I_2	$\begin{bmatrix} -1.000 & -0.013 & 0.021 \\ 0.023 & -0.702 & 0.712 \\ 0.006 & 0.712 & 0.702 \end{bmatrix}$	$\mathbb{D}$ [0.71] -	$\Delta[\Xi_u]$ 0.0855 $\Delta[\mathbb{D}_*]^{\ominus}$ 0.0051
I_3	$\begin{bmatrix} -1.000 & -0.015 & 0.013 \\ -0.020 & 0.707 & -0.707 \\ 0.001 & -0.707 & -0.707 \end{bmatrix}$	$\mathbb{D}$ [0.71] -	$\Delta[\Xi_u]$ 0.0585 $\Delta[\mathbb{D}_*]^{\ominus}$ 0.0003
I_4	$\begin{bmatrix} -0.320 & 0.669 & -0.671 \\ 0.947 & 0.223 & -0.230 \\ -0.004 & -0.709 & -0.705 \end{bmatrix}$	$\mathbb{D}$ [0.71] -	$\Delta[\Xi_u]$ 0.0804 $\Delta[\mathbb{D}_*]^{\ominus}$ 0.0022

(c) Hypothesis H_1 for $n_\theta = 10$.

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\mathbb{D}_*]^{\ominus}$
H_1	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 0.707 & -0.707 \\ 0.000 & 0.707 & 0.707 \end{bmatrix}$	$\mathbb{D}$ [0.71] -	
I_0	$\begin{bmatrix} -0.756 & -0.463 & 0.463 \\ -0.655 & 0.529 & -0.540 \\ 0.006 & -0.711 & -0.703 \end{bmatrix}$	$\mathbb{D}$ [0.71] -	$\Delta[\Xi_u]$ 0.1136 $\Delta[\mathbb{D}_*]^{\ominus}$ 0.0042
I_1	$\begin{bmatrix} 0.120 & 0.707 & -0.697 \\ -0.993 & 0.085 & -0.084 \\ -0.000 & 0.703 & 0.712 \end{bmatrix}$	$\mathbb{D}$ [0.70] -	$\Delta[\Xi_u]$ 0.1045 $\Delta[\mathbb{D}_*]^{\ominus}$ 0.0046
I_2	$\begin{bmatrix} -0.909 & -0.297 & 0.294 \\ 0.418 & -0.642 & 0.643 \\ -0.003 & 0.707 & 0.707 \end{bmatrix}$	$\mathbb{D}$ [0.71] -	$\Delta[\Xi_u]$ 0.0681 $\Delta[\mathbb{D}_*]^{\ominus}$ 0.0003
I_3	$\begin{bmatrix} 0.420 & 0.644 & -0.639 \\ -0.907 & 0.298 & -0.296 \\ -0.000 & 0.705 & 0.710 \end{bmatrix}$	$\mathbb{D}$ [0.70] -	$\Delta[\Xi_u]$ 0.0593 $\Delta[\mathbb{D}_*]^{\ominus}$ 0.0025
I_4	$\begin{bmatrix} -0.259 & 0.683 & -0.683 \\ 0.966 & 0.183 & -0.183 \\ 0.000 & -0.708 & -0.707 \end{bmatrix}$	$\mathbb{D}$ [0.71] -	$\Delta[\Xi_u]$ 0.0810 $\Delta[\mathbb{D}_*]^{\ominus}$ 0.0004

(d) Hypothesis H_1 for $n_\theta = 20$.

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\mathbb{D}_*]^{\ominus}$
H_2	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 0.383 & -0.924 \\ 0.000 & 0.924 & 0.383 \end{bmatrix}$	$\mathbb{D}$ [0.92] -	
I_0	$\begin{bmatrix} 0.870 & 0.189 & -0.455 \\ 0.493 & -0.340 & 0.801 \\ -0.003 & -0.921 & -0.389 \end{bmatrix}$	$\mathbb{D}$ [0.92] -	$\Delta[\Xi_u]$ 0.0820 $\Delta[\mathbb{D}_*]^{\ominus}$ 0.0025
I_1	$\begin{bmatrix} 0.920 & -0.145 & -0.364 \\ 0.391 & 0.406 & 0.826 \\ 0.028 & -0.902 & 0.431 \end{bmatrix}$	$\mathbb{D}$ [0.90] -	$\Delta[\Xi_u]$ 0.1388 $\Delta[\mathbb{D}_*]^{\ominus}$ 0.0213
I_2	$\begin{bmatrix} -0.552 & -0.311 & 0.774 \\ 0.834 & -0.200 & 0.514 \\ -0.005 & 0.929 & 0.370 \end{bmatrix}$	$\mathbb{D}$ [0.93] -	$\Delta[\Xi_u]$ 0.1115 $\Delta[\mathbb{D}_*]^{\ominus}$ 0.0052
I_3	$\begin{bmatrix} -0.997 & 0.031 & 0.077 \\ 0.083 & 0.390 & 0.917 \\ -0.002 & 0.920 & -0.392 \end{bmatrix}$	$\mathbb{D}$ [0.92] -	$\Delta[\Xi_u]$ 0.0777 $\Delta[\mathbb{D}_*]^{\ominus}$ 0.0037
I_4	$\begin{bmatrix} 0.979 & 0.094 & -0.182 \\ -0.204 & 0.376 & -0.904 \\ -0.017 & 0.922 & 0.388 \end{bmatrix}$	$\mathbb{D}$ [0.92] -	$\Delta[\Xi_u]$ 0.0901 $\Delta[\mathbb{D}_*]^{\ominus}$ 0.0020

(e) Hypothesis H_2 for $n_\theta = 10$.

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\mathbb{D}_*]^{\ominus}$
H_2	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 0.383 & -0.924 \\ 0.000 & 0.924 & 0.383 \end{bmatrix}$	$\mathbb{D}$ [0.92] -	
I_0	$\begin{bmatrix} 0.380 & -0.357 & -0.854 \\ -0.925 & -0.150 & -0.349 \\ -0.004 & 0.922 & -0.387 \end{bmatrix}$	$\mathbb{D}$ [0.92] -	$\Delta[\Xi_u]$ 0.1003 $\Delta[\mathbb{D}_*]^{\ominus}$ 0.0019
I_1	$\begin{bmatrix} -0.895 & 0.170 & 0.412 \\ -0.445 & -0.347 & -0.826 \\ 0.002 & -0.923 & 0.386 \end{bmatrix}$	$\mathbb{D}$ [0.92] -	$\Delta[\Xi_u]$ 0.0960 $\Delta[\mathbb{D}_*]^{\ominus}$ 0.0014
I_2	$\begin{bmatrix} -0.813 & -0.210 & 0.544 \\ 0.583 & -0.290 & 0.759 \\ -0.002 & 0.934 & 0.358 \end{bmatrix}$	$\mathbb{D}$ [0.93] -	$\Delta[\Xi_u]$ 0.1040 $\Delta[\mathbb{D}_*]^{\ominus}$ 0.0097
I_3	$\begin{bmatrix} -0.960 & -0.111 & -0.258 \\ -0.281 & 0.380 & 0.881 \\ 0.001 & 0.918 & -0.396 \end{bmatrix}$	$\mathbb{D}$ [0.92] -	$\Delta[\Xi_u]$ 0.0792 $\Delta[\mathbb{D}_*]^{\ominus}$ 0.0057
I_4	$\begin{bmatrix} 0.998 & 0.029 & -0.063 \\ -0.070 & 0.371 & -0.926 \\ -0.004 & 0.928 & 0.372 \end{bmatrix}$	$\mathbb{D}$ [0.93] -	$\Delta[\Xi_u]$ 0.0769 $\Delta[\mathbb{D}_*]^{\ominus}$ 0.0044

(f) Hypothesis H_2 for $n_\theta = 20$.

Table D.61: Non-noisy optimization parameterized with SO(n) on $\mathrm{H}_3^{\ominus}\mathrm{SE}(2)$. Hypotheses H_0 to H_2. True projection matrix and optimization results per iteration I_0 to I_4. Commutativity vector, unnormalized intrinsic conflict and commutativity delta.

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\oplus_*]^{\ominus}$
H_3	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 1.000 & 0.000 \\ 0.000 & 0.000 & 1.000 \end{bmatrix}$	$\oplus$ [0.00]	
		-	
I_0	$\begin{bmatrix} 0.583 & 0.813 & -0.000 \\ 0.813 & -0.583 & 0.011 \\ 0.009 & -0.007 & -1.000 \end{bmatrix}$	$\oplus$ [0.01]	$\Delta[\Xi_u]$ 0.0488
		-	$\Delta[\oplus_*]^{\ominus}$ 0.0112
I_1	$\begin{bmatrix} -0.048 & -0.999 & 0.001 \\ -0.999 & 0.048 & 0.005 \\ -0.005 & -0.001 & -1.000 \end{bmatrix}$	$\oplus$ [0.01]	$\Delta[\Xi_u]$ 0.0522
		-	$\Delta[\oplus_*]^{\ominus}$ 0.0054
I_2	$\begin{bmatrix} -0.146 & 0.989 & 0.004 \\ 0.989 & 0.146 & 0.004 \\ 0.003 & 0.005 & -1.000 \end{bmatrix}$	$\oplus$ [0.01]	$\Delta[\Xi_u]$ 0.0372
		-	$\Delta[\oplus_*]^{\ominus}$ 0.0057
I_3	$\begin{bmatrix} -0.962 & -0.123 & -0.246 \\ -0.058 & 0.965 & -0.255 \\ 0.268 & -0.231 & -0.935 \end{bmatrix}$	$\oplus$ [0.35]	$\Delta[\Xi_u]$ 0.1159
		-	$\Delta[\oplus_*]^{\ominus}$ 0.3537
I_4	$\begin{bmatrix} -0.302 & 0.953 & 0.005 \\ 0.953 & 0.302 & -0.005 \\ -0.006 & 0.003 & -1.000 \end{bmatrix}$	$\oplus$ [0.01]	$\Delta[\Xi_u]$ 0.0397
		-	$\Delta[\oplus_*]^{\ominus}$ 0.0068

(a) Hypothesis H_3 for $n_\theta = 10$.

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\oplus_*]^{\ominus}$
H_3	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 1.000 & 0.000 \\ 0.000 & 0.000 & 1.000 \end{bmatrix}$	$\oplus$ [0.00]	
		-	
I_0	$\begin{bmatrix} 0.636 & 0.771 & 0.003 \\ 0.771 & -0.636 & 0.003 \\ 0.004 & -0.000 & -1.000 \end{bmatrix}$	$\oplus$ [0.00]	$\Delta[\Xi_u]$ 0.0474
		-	$\Delta[\oplus_*]^{\ominus}$ 0.0043
I_1	$\begin{bmatrix} -0.788 & -0.592 & 0.172 \\ -0.551 & 0.801 & 0.234 \\ -0.277 & 0.089 & -0.957 \end{bmatrix}$	$\oplus$ [0.29]	$\Delta[\Xi_u]$ 0.1383
		-	$\Delta[\oplus_*]^{\ominus}$ 0.2906
I_2	$\begin{bmatrix} -0.214 & 0.969 & 0.122 \\ 0.972 & 0.199 & 0.123 \\ 0.095 & 0.145 & -0.985 \end{bmatrix}$	$\oplus$ [0.17]	$\Delta[\Xi_u]$ 0.0894
		-	$\Delta[\oplus_*]^{\ominus}$ 0.1733
I_3	$\begin{bmatrix} -0.209 & -0.978 & 0.003 \\ 0.978 & -0.209 & 0.001 \\ -0.000 & 0.003 & 1.000 \end{bmatrix}$	$\oplus$ [0.00]	$\Delta[\Xi_u]$ 0.0262
		-	$\Delta[\oplus_*]^{\ominus}$ 0.0028
I_4	$\begin{bmatrix} -0.800 & -0.589 & -0.117 \\ -0.552 & 0.798 & -0.244 \\ 0.236 & -0.130 & -0.963 \end{bmatrix}$	$\oplus$ [0.27]	$\Delta[\Xi_u]$ 0.1031
		-	$\Delta[\oplus_*]^{\ominus}$ 0.2700

(b) Hypothesis H_3 for $n_\theta = 20$.

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\oplus_*]^{\ominus}$
H_4	$\begin{bmatrix} 0.000 & 0.000 & 1.000 \\ 1.000 & 0.000 & 0.000 \\ 0.000 & 1.000 & 0.000 \end{bmatrix}$	$\oplus$ [1.00]	
		-	
I_0	$\begin{bmatrix} -0.983 & -0.010 & 0.181 \\ 0.181 & -0.044 & 0.982 \\ -0.002 & 0.999 & 0.045 \end{bmatrix}$	$\oplus$ [1.00]	$\Delta[\Xi_u]$ 0.0897
		-	$\Delta[\oplus_*]^{\ominus}$ 0.0010
I_1	$\begin{bmatrix} 0.780 & -0.007 & 0.626 \\ 0.626 & 0.006 & -0.780 \\ 0.002 & 1.000 & 0.009 \end{bmatrix}$	$\oplus$ [1.00]	$\Delta[\Xi_u]$ 0.0689
		-	$\Delta[\oplus_*]^{\ominus}$ 0.0000
I_2	$\begin{bmatrix} -0.109 & -0.038 & 0.993 \\ 0.994 & -0.002 & 0.109 \\ -0.002 & 0.999 & 0.038 \end{bmatrix}$	$\oplus$ [1.00]	$\Delta[\Xi_u]$ 0.0882
		-	$\Delta[\oplus_*]^{\ominus}$ 0.0007
I_3	$\begin{bmatrix} -0.998 & -0.060 & -0.036 \\ -0.053 & 0.316 & 0.947 \\ -0.045 & 0.947 & -0.318 \end{bmatrix}$	$\oplus$ [0.95]	$\Delta[\Xi_u]$ 0.1251
		-	$\Delta[\oplus_*]^{\ominus}$ 0.0520
I_4	$\begin{bmatrix} 0.229 & 0.023 & -0.973 \\ 0.973 & -0.004 & 0.229 \\ 0.001 & -1.000 & -0.023 \end{bmatrix}$	$\oplus$ [1.00]	$\Delta[\Xi_u]$ 0.0725
		-	$\Delta[\oplus_*]^{\ominus}$ 0.0003

(c) Hypothesis H_4 for $n_\theta = 10$.

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\oplus_*]^{\ominus}$
H_4	$\begin{bmatrix} 0.000 & 0.000 & 1.000 \\ 1.000 & 0.000 & 0.000 \\ 0.000 & 1.000 & 0.000 \end{bmatrix}$	$\oplus$ [1.00]	
		-	
I_0	$\begin{bmatrix} -0.715 & 0.016 & 0.699 \\ 0.699 & 0.021 & 0.715 \\ -0.003 & 1.000 & -0.026 \end{bmatrix}$	$\oplus$ [1.00]	$\Delta[\Xi_u]$ 0.0882
		-	$\Delta[\oplus_*]^{\ominus}$ 0.0003
I_1	$\begin{bmatrix} 0.597 & 0.024 & 0.802 \\ 0.802 & -0.023 & -0.597 \\ 0.004 & 0.999 & -0.033 \end{bmatrix}$	$\oplus$ [1.00]	$\Delta[\Xi_u]$ 0.0652
		-	$\Delta[\oplus_*]^{\ominus}$ 0.0005
I_2	$\begin{bmatrix} 0.937 & -0.011 & -0.348 \\ -0.348 & -0.039 & -0.937 \\ -0.003 & 0.999 & -0.041 \end{bmatrix}$	$\oplus$ [1.00]	$\Delta[\Xi_u]$ 0.0883
		-	$\Delta[\oplus_*]^{\ominus}$ 0.0008
I_3	$\begin{bmatrix} -0.182 & -0.008 & 0.983 \\ -0.983 & 0.001 & -0.182 \\ 0.000 & -1.000 & -0.008 \end{bmatrix}$	$\oplus$ [1.00]	$\Delta[\Xi_u]$ 0.0865
		-	$\Delta[\oplus_*]^{\ominus}$ 0.0000
I_4	$\begin{bmatrix} 0.021 & -0.007 & -1.000 \\ 1.000 & 0.007 & 0.021 \\ 0.007 & -1.000 & 0.008 \end{bmatrix}$	$\oplus$ [1.00]	$\Delta[\Xi_u]$ 0.0735
		-	$\Delta[\oplus_*]^{\ominus}$ 0.0000

(d) Hypothesis H_4 for $n_\theta = 20$.

Table D.62: Non-noisy optimization parameterized with SO(n) on $\mathrm{H}_3^{\ominus}\mathrm{SE}(2)$. Hypotheses H_3 to H_4. True projection matrix and optimization results per iteration I_0 to I_4. Commutativity vector, unnormalized intrinsic conflict and commutativity delta.

Id	Hypothesis, $\Delta[\oplus_*]^{\ominus}$				
	H_0	H_1	H_2	H_3	H_4
I_0	0.0092	0.0002	0.0025	0.0112	0.0010
I_1	0.0857	0.0019	0.0213	0.0054	0.0000
I_2	0.0086	0.0051	0.0052	0.0057	0.0007
I_3	0.0025	0.0003	0.0037	0.3537	0.0520
I_4	0.0069	0.0022	0.0020	0.0068	0.0003
μ	0.0226	0.0019	0.0069	0.0766	0.0108
σ	0.0316	0.0018	0.0073	0.1386	0.0206
$\mu = 0.0238$, $\sigma = 0.0713$					

(a) Evaluation for $n_\theta = 10$.

Id	Hypothesis, $\Delta[\oplus_*]^{\ominus}$				
	H_0	H_1	H_2	H_3	H_4
I_0	0.0033	0.0042	0.0019	0.0043	0.0003
I_1	0.0037	0.0046	0.0014	0.2906	0.0005
I_2	0.0027	0.0003	0.0097	0.1733	0.0008
I_3	0.0108	0.0025	0.0057	0.0028	0.0000
I_4	0.0045	0.0004	0.0044	0.2700	0.0000
μ	0.0050	0.0024	0.0046	0.1482	0.0003
σ	0.0030	0.0018	0.0030	0.1246	0.0003
$\mu = 0.0321$, $\sigma = 0.0822$					

(b) Evaluation for $n_\theta = 20$.

Table D.63: Non-noisy optimization parameterized with SO(n) on $\mathrm{H}_3^{\ominus}\mathrm{SE}(2)$. Commutativity delta $\Delta[\oplus_*]^{\ominus}$ per iteration I_0 to I_4. Mean and standard deviation of $\Delta[\oplus_*]^{\ominus}$ per hypothesis H_0 to H_4 and over all hypotheses.

D.11.2 Non-Noisy Optimization parameterized with GL(n,R) on $\mathrm{H}_3^{\ominus}\mathrm{SE}(2)$

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\mathbb{D}_*]^{\ominus}$
H_0	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 0.924 & -0.383 \\ 0.000 & 0.383 & 0.924 \end{bmatrix}$	$\mathbb{D}$ $\begin{bmatrix} 0.38 \end{bmatrix}$ -	
I_0	$\begin{bmatrix} 2.799 & -0.705 & 0.258 \\ 1.281 & -0.259 & 0.653 \\ -0.015 & -0.408 & 0.225 \end{bmatrix}$	$\mathbb{D}$ $\begin{bmatrix} 1.27 \end{bmatrix}$ -	$\Delta[\Xi_u]$ 0.1789 $\Delta[\mathbb{D}_*]^{\ominus}$ 0.8855
I_1	$\begin{bmatrix} 6.695 & 2.240 & -7.317 \\ 3.437 & 1.505 & -3.494 \\ -0.602 & 0.697 & 4.027 \end{bmatrix}$	$\mathbb{D}$ $\begin{bmatrix} 6.86 \end{bmatrix}$ -	$\Delta[\Xi_u]$ 0.1985 $\Delta[\mathbb{D}_*]^{\ominus}$ 6.4784
I_2	$\begin{bmatrix} 1.508 & 2.460 & -0.225 \\ 0.973 & 3.245 & -0.648 \\ 0.576 & -0.156 & 0.440 \end{bmatrix}$	$\mathbb{D}$ $\begin{bmatrix} 2.61 \end{bmatrix}$ -	$\Delta[\Xi_u]$ 0.1620 $\Delta[\mathbb{D}_*]^{\ominus}$ 2.2289
I_3	$\begin{bmatrix} 0.238 & -0.016 & 0.066 \\ 0.218 & 0.157 & -0.117 \\ 0.052 & 0.229 & 0.371 \end{bmatrix}$	$\mathbb{D}$ $\begin{bmatrix} 0.07 \end{bmatrix}$ -	$\Delta[\Xi_u]$ 0.1226 $\Delta[\mathbb{D}_*]^{\ominus}$ 0.3132
I_4	$\begin{bmatrix} 1.295 & 0.374 & 0.313 \\ 1.428 & 1.414 & -0.519 \\ 0.008 & 0.180 & 0.158 \end{bmatrix}$	$\mathbb{D}$ $\begin{bmatrix} 0.34 \end{bmatrix}$ -	$\Delta[\Xi_u]$ 0.0990 $\Delta[\mathbb{D}_*]^{\ominus}$ 0.0473

(a) Hypothesis H_0 for $n_\theta = 10$.

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\mathbb{D}_*]^{\ominus}$
H_0	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 0.924 & -0.383 \\ 0.000 & 0.383 & 0.924 \end{bmatrix}$	$\mathbb{D}$ $\begin{bmatrix} 0.38 \end{bmatrix}$ -	
I_0	$\begin{bmatrix} 3.385 & -0.269 & 0.766 \\ 2.232 & 0.328 & 1.115 \\ -0.195 & -0.565 & 0.146 \end{bmatrix}$	$\mathbb{D}$ $\begin{bmatrix} 2.30 \end{bmatrix}$ -	$\Delta[\Xi_u]$ 0.1873 $\Delta[\mathbb{D}_*]^{\ominus}$ 1.9190
I_1	$\begin{bmatrix} 1.503 & 0.260 & 0.581 \\ -0.115 & 1.046 & 0.064 \\ 0.373 & 0.072 & 0.462 \end{bmatrix}$	$\mathbb{D}$ $\begin{bmatrix} 0.40 \end{bmatrix}$ -	$\Delta[\Xi_u]$ 0.1528 $\Delta[\mathbb{D}_*]^{\ominus}$ 0.0162
I_2	$\begin{bmatrix} 2.023 & -2.010 & -0.622 \\ -0.607 & 2.029 & -0.944 \\ -0.001 & 0.382 & 0.278 \end{bmatrix}$	$\mathbb{D}$ $\begin{bmatrix} 0.81 \end{bmatrix}$ -	$\Delta[\Xi_u]$ 0.0910 $\Delta[\mathbb{D}_*]^{\ominus}$ 0.4228
I_3	$\begin{bmatrix} 0.064 & 0.342 & 0.153 \\ -0.254 & 0.473 & -0.128 \\ -0.196 & -0.298 & 0.372 \end{bmatrix}$	$\mathbb{D}$ $\begin{bmatrix} 0.18 \end{bmatrix}$ -	$\Delta[\Xi_u]$ 0.1364 $\Delta[\mathbb{D}_*]^{\ominus}$ 0.2075
I_4	$\begin{bmatrix} 0.654 & 0.577 & 0.201 \\ 0.414 & 1.433 & -0.551 \\ -0.120 & 0.160 & 0.133 \end{bmatrix}$	$\mathbb{D}$ $\begin{bmatrix} 0.30 \end{bmatrix}$ -	$\Delta[\Xi_u]$ 0.1035 $\Delta[\mathbb{D}_*]^{\ominus}$ 0.0873

(b) Hypothesis H_0 for $n_\theta = 20$.

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\mathbb{D}_*]^{\ominus}$
H_1	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 0.707 & -0.707 \\ 0.000 & 0.707 & 0.707 \end{bmatrix}$	$\mathbb{D}$ $\begin{bmatrix} 0.71 \end{bmatrix}$ -	
I_0	$\begin{bmatrix} 3.739 & -0.722 & 0.964 \\ 1.856 & -0.073 & 0.753 \\ 0.002 & -0.696 & 0.486 \end{bmatrix}$	$\mathbb{D}$ $\begin{bmatrix} 2.91 \end{bmatrix}$ -	$\Delta[\Xi_u]$ 0.1446 $\Delta[\mathbb{D}_*]^{\ominus}$ 2.1988
I_1	$\begin{bmatrix} 6.712 & 2.226 & -3.518 \\ 3.098 & 1.442 & -1.084 \\ -0.814 & 0.202 & 3.164 \end{bmatrix}$	$\mathbb{D}$ $\begin{bmatrix} 3.64 \end{bmatrix}$ -	$\Delta[\Xi_u]$ 0.2187 $\Delta[\mathbb{D}_*]^{\ominus}$ 2.9323
I_2	$\begin{bmatrix} 0.899 & 1.130 & -0.047 \\ 0.405 & 1.565 & -0.473 \\ 0.111 & -0.605 & 0.445 \end{bmatrix}$	$\mathbb{D}$ $\begin{bmatrix} 0.79 \end{bmatrix}$ -	$\Delta[\Xi_u]$ 0.1206 $\Delta[\mathbb{D}_*]^{\ominus}$ 0.0818
I_3	$\begin{bmatrix} -0.171 & 0.412 & 0.159 \\ -0.421 & 0.395 & -0.290 \\ 0.002 & -0.717 & 0.351 \end{bmatrix}$	$\mathbb{D}$ $\begin{bmatrix} 0.32 \end{bmatrix}$ -	$\Delta[\Xi_u]$ 0.0773 $\Delta[\mathbb{D}_*]^{\ominus}$ 0.3825
I_4	$\begin{bmatrix} 0.135 & -1.102 & -0.013 \\ 1.268 & 2.134 & 0.916 \\ -0.006 & -0.707 & 0.201 \end{bmatrix}$	$\mathbb{D}$ $\begin{bmatrix} 0.89 \end{bmatrix}$ -	$\Delta[\Xi_u]$ 0.1076 $\Delta[\mathbb{D}_*]^{\ominus}$ 0.1816

(c) Hypothesis H_1 for $n_\theta = 10$.

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\mathbb{D}_*]^{\ominus}$
H_1	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 0.707 & -0.707 \\ 0.000 & 0.707 & 0.707 \end{bmatrix}$	$\mathbb{D}$ $\begin{bmatrix} 0.71 \end{bmatrix}$ -	
I_0	$\begin{bmatrix} 2.874 & -0.484 & 0.002 \\ 1.440 & -0.125 & 0.447 \\ 0.120 & -0.289 & 0.191 \end{bmatrix}$	$\mathbb{D}$ $\begin{bmatrix} 0.87 \end{bmatrix}$ -	$\Delta[\Xi_u]$ 0.1474 $\Delta[\mathbb{D}_*]^{\ominus}$ 0.1648
I_1	$\begin{bmatrix} 0.711 & 0.299 & 0.322 \\ -0.229 & 0.809 & -0.254 \\ -0.005 & 0.707 & 0.708 \end{bmatrix}$	$\mathbb{D}$ $\begin{bmatrix} 0.53 \end{bmatrix}$ -	$\Delta[\Xi_u]$ 0.1048 $\Delta[\mathbb{D}_*]^{\ominus}$ 0.1787
I_2	$\begin{bmatrix} 0.861 & 1.349 & -0.399 \\ 0.939 & 3.677 & -1.242 \\ 0.220 & -0.293 & 0.386 \end{bmatrix}$	$\mathbb{D}$ $\begin{bmatrix} 1.21 \end{bmatrix}$ -	$\Delta[\Xi_u]$ 0.1231 $\Delta[\mathbb{D}_*]^{\ominus}$ 0.5077
I_3	$\begin{bmatrix} 0.236 & 0.856 & 0.367 \\ -0.680 & 0.535 & -0.145 \\ -0.001 & -0.702 & 0.331 \end{bmatrix}$	$\mathbb{D}$ $\begin{bmatrix} 0.51 \end{bmatrix}$ -	$\Delta[\Xi_u]$ 0.0780 $\Delta[\mathbb{D}_*]^{\ominus}$ 0.2013
I_4	$\begin{bmatrix} 0.995 & 0.955 & 0.473 \\ -0.084 & 1.505 & -0.930 \\ 0.005 & 0.715 & 0.209 \end{bmatrix}$	$\mathbb{D}$ $\begin{bmatrix} 0.71 \end{bmatrix}$ -	$\Delta[\Xi_u]$ 0.0902 $\Delta[\mathbb{D}_*]^{\ominus}$ 0.0029

(d) Hypothesis H_1 for $n_\theta = 20$.

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\mathbb{D}_*]^{\ominus}$
H_2	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 0.383 & -0.924 \\ 0.000 & 0.924 & 0.383 \end{bmatrix}$	$\mathbb{D}$ $\begin{bmatrix} 0.92 \end{bmatrix}$ -	
I_0	$\begin{bmatrix} 3.359 & -0.496 & 0.574 \\ 1.618 & -0.294 & 0.767 \\ -0.050 & -0.514 & 0.232 \end{bmatrix}$	$\mathbb{D}$ $\begin{bmatrix} 1.95 \end{bmatrix}$ -	$\Delta[\Xi_u]$ 0.1254 $\Delta[\mathbb{D}_*]^{\ominus}$ 1.0216
I_1	$\begin{bmatrix} 6.172 & 1.911 & -4.543 \\ 3.032 & 1.411 & -1.554 \\ -0.148 & 0.727 & 3.064 \end{bmatrix}$	$\mathbb{D}$ $\begin{bmatrix} 5.35 \end{bmatrix}$ -	$\Delta[\Xi_u]$ 0.1608 $\Delta[\mathbb{D}_*]^{\ominus}$ 4.4226
I_2	$\begin{bmatrix} 1.362 & 0.523 & 0.710 \\ 0.870 & 0.938 & 0.639 \\ 0.220 & -0.669 & 0.252 \end{bmatrix}$	$\mathbb{D}$ $\begin{bmatrix} 1.29 \end{bmatrix}$ -	$\Delta[\Xi_u]$ 0.1974 $\Delta[\mathbb{D}_*]^{\ominus}$ 0.3695
I_3	$\begin{bmatrix} 0.286 & 0.332 & 0.374 \\ -0.692 & 0.607 & -0.180 \\ 0.002 & -0.922 & 0.892 \end{bmatrix}$	$\mathbb{D}$ $\begin{bmatrix} 0.69 \end{bmatrix}$ -	$\Delta[\Xi_u]$ 0.0884 $\Delta[\mathbb{D}_*]^{\ominus}$ 0.2347
I_4	$\begin{bmatrix} 2.715 & 0.831 & 0.473 \\ 2.235 & 1.356 & 0.211 \\ 0.188 & -0.326 & 0.338 \end{bmatrix}$	$\mathbb{D}$ $\begin{bmatrix} 1.43 \end{bmatrix}$ -	$\Delta[\Xi_u]$ 0.1369 $\Delta[\mathbb{D}_*]^{\ominus}$ 0.5097

(e) Hypothesis H_2 for $n_\theta = 10$.

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\mathbb{D}_*]^{\ominus}$
H_2	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 0.383 & -0.924 \\ 0.000 & 0.924 & 0.383 \end{bmatrix}$	$\mathbb{D}$ $\begin{bmatrix} 0.92 \end{bmatrix}$ -	
I_0	$\begin{bmatrix} 4.046 & -1.201 & 0.338 \\ 2.042 & -0.330 & 0.802 \\ -0.151 & -0.364 & 0.074 \end{bmatrix}$	$\mathbb{D}$ $\begin{bmatrix} 1.84 \end{bmatrix}$ -	$\Delta[\Xi_u]$ 0.1199 $\Delta[\mathbb{D}_*]^{\ominus}$ 0.9120
I_1	$\begin{bmatrix} 0.322 & 0.654 & -0.150 \\ 0.620 & 2.961 & -0.228 \\ 0.317 & 0.004 & 0.149 \end{bmatrix}$	$\mathbb{D}$ $\begin{bmatrix} 0.96 \end{bmatrix}$ -	$\Delta[\Xi_u]$ 0.1313 $\Delta[\mathbb{D}_*]^{\ominus}$ 0.0333
I_2	$\begin{bmatrix} 0.942 & -0.281 & 0.468 \\ 0.510 & 1.469 & -0.529 \\ -0.005 & -0.917 & 0.878 \end{bmatrix}$	$\mathbb{D}$ $\begin{bmatrix} 0.98 \end{bmatrix}$ -	$\Delta[\Xi_u]$ 0.1314 $\Delta[\mathbb{D}_*]^{\ominus}$ 0.0562
I_3	$\begin{bmatrix} 1.078 & 0.340 & 0.258 \\ 0.709 & 1.574 & 0.156 \\ -0.002 & -0.925 & 0.555 \end{bmatrix}$	$\mathbb{D}$ $\begin{bmatrix} 1.19 \end{bmatrix}$ -	$\Delta[\Xi_u]$ 0.0944 $\Delta[\mathbb{D}_*]^{\ominus}$ 0.2668
I_4	$\begin{bmatrix} 2.454 & 0.933 & 0.420 \\ 2.552 & 1.698 & 0.190 \\ 0.180 & -0.335 & 0.390 \end{bmatrix}$	$\mathbb{D}$ $\begin{bmatrix} 1.52 \end{bmatrix}$ -	$\Delta[\Xi_u]$ 0.1393 $\Delta[\mathbb{D}_*]^{\ominus}$ 0.6002

(f) Hypothesis H_2 for $n_\theta = 20$.

Table D.64: Non-noisy optimization parameterized with GL(n,R) on $\mathrm{H}_3^{\ominus}\mathrm{SE}(2)$. Hypotheses H_0 to H_2. True projection matrix and optimization results per iteration I_0 to I_4. Commutativity vector, unnormalized intrinsic conflict and commutativity delta.

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\circledcirc_*]^{\ominus}$
H_3	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 1.000 & 0.000 \\ 0.000 & 0.000 & 1.000 \end{bmatrix}$	ⓘ [0.00]	
		-	
I_0	$\begin{bmatrix} 3.108 & -0.322 & 0.095 \\ 1.998 & -0.028 & 0.598 \\ -0.136 & -0.358 & 0.159 \end{bmatrix}$	ⓘ [1.36]	$\Delta[\Xi_u]$ 0.1556
		-	$\Delta[\circledcirc_*]^{\ominus}$ 1.3606
I_1	$\begin{bmatrix} 7.673 & 2.756 & -4.705 \\ 4.615 & 2.015 & -2.383 \\ -0.304 & 0.653 & 3.636 \end{bmatrix}$	ⓘ [6.88]	$\Delta[\Xi_u]$ 0.1717
		-	$\Delta[\circledcirc_*]^{\ominus}$ 6.8813
I_2	$\begin{bmatrix} 3.753 & 2.514 & 0.694 \\ 1.073 & 2.024 & -0.280 \\ 0.743 & -0.477 & 0.807 \end{bmatrix}$	ⓘ [4.18]	$\Delta[\Xi_u]$ 0.1097
		-	$\Delta[\circledcirc_*]^{\ominus}$ 4.1776
I_3	$\begin{bmatrix} 0.223 & 0.121 & 0.191 \\ 0.060 & 0.622 & -0.359 \\ -0.000 & 0.000 & 0.611 \end{bmatrix}$	ⓘ [0.00]	$\Delta[\Xi_u]$ 0.0260
		-	$\Delta[\circledcirc_*]^{\ominus}$ 0.0001
I_4	$\begin{bmatrix} 1.015 & 1.389 & -0.061 \\ 2.153 & 4.380 & -1.099 \\ -0.220 & 0.037 & 0.136 \end{bmatrix}$	ⓘ [1.10]	$\Delta[\Xi_u]$ 0.1069
		-	$\Delta[\circledcirc_*]^{\ominus}$ 1.0967

(a) Hypothesis H_0 for $n_\theta = 10$.

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\circledcirc_*]^{\ominus}$
H_3	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 1.000 & 0.000 \\ 0.000 & 0.000 & 1.000 \end{bmatrix}$	ⓘ [0.00]	
		-	
I_0	$\begin{bmatrix} 1.055 & 1.697 & -0.033 \\ -0.899 & -0.605 & 0.336 \\ -0.004 & -0.003 & 0.864 \end{bmatrix}$	ⓘ [0.00]	$\Delta[\Xi_u]$ 0.0478
		-	$\Delta[\circledcirc_*]^{\ominus}$ 0.0036
I_1	$\begin{bmatrix} 0.329 & 0.553 & -0.297 \\ 0.478 & 2.743 & -0.356 \\ 0.235 & -0.089 & 0.174 \end{bmatrix}$	ⓘ [0.71]	$\Delta[\Xi_u]$ 0.1335
		-	$\Delta[\circledcirc_*]^{\ominus}$ 0.7064
I_2	$\begin{bmatrix} 1.704 & -0.774 & 0.068 \\ 1.234 & 2.143 & -1.699 \\ -0.016 & 0.073 & 0.077 \end{bmatrix}$	ⓘ [0.17]	$\Delta[\Xi_u]$ 0.0458
		-	$\Delta[\circledcirc_*]^{\ominus}$ 0.1662
I_3	$\begin{bmatrix} 0.326 & 0.031 & 0.136 \\ 0.284 & 0.447 & -0.058 \\ -0.000 & -0.000 & 0.408 \end{bmatrix}$	ⓘ [0.00]	$\Delta[\Xi_u]$ 0.0260
		-	$\Delta[\circledcirc_*]^{\ominus}$ 0.0001
I_4	$\begin{bmatrix} 0.680 & 0.383 & 0.398 \\ 1.150 & 2.853 & -0.136 \\ -0.001 & -0.001 & 0.329 \end{bmatrix}$	ⓘ [0.00]	$\Delta[\Xi_u]$ 0.0372
		-	$\Delta[\circledcirc_*]^{\ominus}$ 0.0028

(b) Hypothesis H_0 for $n_\theta = 20$.

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\circledcirc_*]^{\ominus}$
H_4	$\begin{bmatrix} 0.000 & 0.000 & 1.000 \\ 1.000 & 0.000 & 0.000 \\ 0.000 & 1.000 & 0.000 \end{bmatrix}$	ⓘ [1.00]	
		-	
I_0	$\begin{bmatrix} 3.239 & -0.755 & 0.654 \\ 1.421 & -0.292 & 0.645 \\ 0.034 & -0.337 & 0.276 \end{bmatrix}$	ⓘ [1.17]	$\Delta[\Xi_u]$ 0.1452
		-	$\Delta[\circledcirc_*]^{\ominus}$ 0.1664
I_1	$\begin{bmatrix} 7.412 & 3.353 & -4.228 \\ 2.882 & 1.680 & -1.140 \\ -0.723 & 0.422 & 4.301 \end{bmatrix}$	ⓘ [6.06]	$\Delta[\Xi_u]$ 0.1464
		-	$\Delta[\circledcirc_*]^{\ominus}$ 5.0573
I_2	$\begin{bmatrix} 1.303 & 2.732 & -0.494 \\ 1.387 & 4.395 & -0.878 \\ 0.004 & -1.050 & 0.441 \end{bmatrix}$	ⓘ [2.02]	$\Delta[\Xi_u]$ 0.1339
		-	$\Delta[\circledcirc_*]^{\ominus}$ 1.0209
I_3	$\begin{bmatrix} 0.421 & -0.109 & -0.588 \\ 0.264 & 0.684 & 0.836 \\ 0.008 & 0.995 & 2.448 \end{bmatrix}$	ⓘ [0.49]	$\Delta[\Xi_u]$ 0.1106
		-	$\Delta[\circledcirc_*]^{\ominus}$ 0.5073
I_4	$\begin{bmatrix} 0.940 & 1.134 & -0.026 \\ 2.261 & 3.665 & -0.993 \\ -0.206 & 0.012 & 0.128 \end{bmatrix}$	ⓘ [0.82]	$\Delta[\Xi_u]$ 0.1253
		-	$\Delta[\circledcirc_*]^{\ominus}$ 0.1802

(c) Hypothesis H_1 for $n_\theta = 10$.

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\circledcirc_*]^{\ominus}$
H_4	$\begin{bmatrix} 0.000 & 0.000 & 1.000 \\ 1.000 & 0.000 & 0.000 \\ 0.000 & 1.000 & 0.000 \end{bmatrix}$	ⓘ [1.00]	
		-	
I_0	$\begin{bmatrix} 3.821 & -0.149 & 0.392 \\ 1.881 & -0.129 & 0.784 \\ -0.103 & -0.560 & 0.340 \end{bmatrix}$	ⓘ [2.40]	$\Delta[\Xi_u]$ 0.1524
		-	$\Delta[\circledcirc_*]^{\ominus}$ 1.4033
I_1	$\begin{bmatrix} 1.066 & 1.176 & -0.383 \\ 1.628 & 2.922 & -0.712 \\ 0.360 & -0.287 & 0.379 \end{bmatrix}$	ⓘ [1.69]	$\Delta[\Xi_u]$ 0.1365
		-	$\Delta[\circledcirc_*]^{\ominus}$ 0.6852
I_2	$\begin{bmatrix} 1.050 & 1.614 & -0.343 \\ 1.087 & 3.377 & -0.634 \\ 0.002 & -1.000 & 0.401 \end{bmatrix}$	ⓘ [1.52]	$\Delta[\Xi_u]$ 0.1145
		-	$\Delta[\circledcirc_*]^{\ominus}$ 0.5170
I_3	$\begin{bmatrix} 5.706 & 1.926 & 0.341 \\ 4.739 & 2.992 & 0.200 \\ -0.162 & -0.627 & 0.207 \end{bmatrix}$	ⓘ [4.11]	$\Delta[\Xi_u]$ 0.1301
		-	$\Delta[\circledcirc_*]^{\ominus}$ 3.1062
I_4	$\begin{bmatrix} 0.803 & 1.277 & 0.036 \\ 2.075 & 5.068 & -0.528 \\ -0.278 & 0.004 & 0.047 \end{bmatrix}$	ⓘ [1.46]	$\Delta[\Xi_u]$ 0.1198
		-	$\Delta[\circledcirc_*]^{\ominus}$ 0.4619

(d) Hypothesis H_1 for $n_\theta = 20$.

Table D.65: Non-noisy optimization parameterized with GL(n,R) on $\mathrm{H}_3^{\ominus}\mathrm{SE}(2)$. Hypotheses H_3 to H_4. True projection matrix and optimization results per iteration I_0 to I_4. Commutativity vector, unnormalized intrinsic conflict and commutativity delta.

Id	Hypothesis, $\Delta[\circledcirc_*]^{\ominus}$				
	H_0	H_1	H_2	H_3	H_4
I_0	0.8855	2.1988	1.0216	1.3606	0.1664
I_1	6.4784	2.9323	4.4226	6.8813	5.0573
I_2	2.2289	0.0818	0.3695	4.1776	1.0209
I_3	0.3132	0.3825	0.2347	0.0001	0.5073
I_4	0.0473	0.1816	0.5097	1.0967	0.1802
μ	1.9907	1.1554	1.3116	2.7033	1.3864
σ	2.3668	1.1785	1.5781	2.5028	1.8615
$\mu = 1.7095$, $\sigma = 2.0840$					

(a) Evaluation for $n_\theta = 10$.

Id	Hypothesis, $\Delta[\circledcirc_*]^{\ominus}$				
	H_0	H_1	H_2	H_3	H_4
I_0	1.9190	0.1648	0.9120	0.0036	1.4033
I_1	0.0162	0.1787	0.0333	0.7064	0.6852
I_2	0.4228	0.5077	0.0562	0.1662	0.5170
I_3	0.2075	0.2013	0.2668	0.0001	3.1062
I_4	0.0873	0.0029	0.6002	0.0028	0.4619
μ	0.5306	0.2111	0.3737	0.1758	1.2347
σ	0.7078	0.1641	0.3374	0.2728	0.9945
$\mu = 0.5052$, $\sigma = 0.7145$					

(b) Evaluation for $n_\theta = 20$.

Table D.66: Non-noisy optimization parameterized with SO(n) on $\mathrm{H}_3^{\ominus}\mathrm{SE}(2)$. Commutativity delta $\Delta[\circledcirc_*]^{\ominus}$ per iteration I_0 to I_4. Mean and standard deviation of $\Delta[\circledcirc_*]^{\ominus}$ per hypothesis H_0 to H_4 and over all hypotheses.

D.11.3 Noisy Optimization parameterized with SO(n) on $\mathrm{H}_3^{\ominus}\mathrm{SE}(2)$

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\mathbb{D}_*]^{\ominus}$
H_0	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 0.924 & -0.383 \\ 0.000 & 0.383 & 0.924 \end{bmatrix}$	$\mathbb{D}$ $[0.38]$ -	
I_0	$\begin{bmatrix} 0.591 & 0.745 & -0.308 \\ 0.807 & -0.549 & 0.220 \\ -0.005 & -0.379 & -0.926 \end{bmatrix}$	$\mathbb{D}$ $[0.38]$ -	$\Delta[\Xi_u]$ 0.1550 $\Delta[\mathbb{D}_*]^{\ominus}$ 0.0039
I_1	$\begin{bmatrix} -0.887 & -0.437 & 0.146 \\ -0.459 & 0.814 & -0.356 \\ 0.036 & -0.383 & -0.923 \end{bmatrix}$	$\mathbb{D}$ $[0.38]$ -	$\Delta[\Xi_u]$ 0.1402 $\Delta[\mathbb{D}_*]^{\ominus}$ 0.0019
I_2	$\begin{bmatrix} 0.170 & 0.908 & -0.384 \\ -0.985 & 0.170 & -0.033 \\ 0.035 & 0.383 & 0.923 \end{bmatrix}$	$\mathbb{D}$ $[0.39]$ -	$\Delta[\Xi_u]$ 0.1320 $\Delta[\mathbb{D}_*]^{\ominus}$ 0.0024
I_3	$\begin{bmatrix} -0.792 & 0.568 & -0.224 \\ 0.611 & 0.741 & -0.279 \\ 0.008 & -0.357 & -0.934 \end{bmatrix}$	$\mathbb{D}$ $[0.36]$ -	$\Delta[\Xi_u]$ 0.1111 $\Delta[\mathbb{D}_*]^{\ominus}$ 0.0254
I_4	$\begin{bmatrix} 0.997 & -0.078 & -0.012 \\ 0.067 & 0.921 & -0.385 \\ 0.041 & 0.383 & 0.923 \end{bmatrix}$	$\mathbb{D}$ $[0.38]$ -	$\Delta[\Xi_u]$ 0.1355 $\Delta[\mathbb{D}_*]^{\ominus}$ 0.0021

(a) Hypothesis H_0 for $n_\theta = 10$.

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\mathbb{D}_*]^{\ominus}$
H_0	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 0.924 & -0.383 \\ 0.000 & 0.383 & 0.924 \end{bmatrix}$	$\mathbb{D}$ $[0.38]$ -	
I_0	$\begin{bmatrix} -0.908 & -0.386 & 0.160 \\ -0.418 & 0.844 & -0.336 \\ -0.006 & -0.372 & -0.928 \end{bmatrix}$	$\mathbb{D}$ $[0.37]$ -	$\Delta[\Xi_u]$ 0.1481 $\Delta[\mathbb{D}_*]^{\ominus}$ 0.0107
I_1	$\begin{bmatrix} 0.746 & -0.605 & 0.278 \\ -0.665 & -0.701 & 0.258 \\ 0.038 & -0.377 & -0.925 \end{bmatrix}$	$\mathbb{D}$ $[0.38]$ -	$\Delta[\Xi_u]$ 0.1432 $\Delta[\mathbb{D}_*]^{\ominus}$ 0.0035
I_2	$\begin{bmatrix} -0.839 & -0.494 & 0.225 \\ -0.543 & 0.777 & -0.317 \\ -0.018 & -0.389 & -0.921 \end{bmatrix}$	$\mathbb{D}$ $[0.39]$ -	$\Delta[\Xi_u]$ 0.1274 $\Delta[\mathbb{D}_*]^{\ominus}$ 0.0067
I_3	$\begin{bmatrix} -0.720 & 0.639 & -0.271 \\ 0.694 & 0.679 & -0.240 \\ 0.030 & -0.361 & -0.932 \end{bmatrix}$	$\mathbb{D}$ $[0.36]$ -	$\Delta[\Xi_u]$ 0.1074 $\Delta[\mathbb{D}_*]^{\ominus}$ 0.0207
I_4	$\begin{bmatrix} -0.049 & 0.927 & -0.372 \\ 0.999 & 0.039 & -0.034 \\ -0.017 & -0.373 & -0.928 \end{bmatrix}$	$\mathbb{D}$ $[0.37]$ -	$\Delta[\Xi_u]$ 0.1368 $\Delta[\mathbb{D}_*]^{\ominus}$ 0.0094

(b) Hypothesis H_0 for $n_\theta = 20$.

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\mathbb{D}_*]^{\ominus}$
H_1	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 0.707 & -0.707 \\ 0.000 & 0.707 & 0.707 \end{bmatrix}$	$\mathbb{D}$ $[0.71]$ -	
I_0	$\begin{bmatrix} -0.805 & -0.398 & 0.441 \\ -0.594 & 0.544 & -0.592 \\ -0.004 & -0.738 & -0.675 \end{bmatrix}$	$\mathbb{D}$ $[0.74]$ -	$\Delta[\Xi_u]$ 0.1402 $\Delta[\mathbb{D}_*]^{\ominus}$ 0.0311
I_1	$\begin{bmatrix} -0.815 & 0.563 & 0.134 \\ 0.480 & 0.787 & -0.387 \\ -0.323 & -0.251 & -0.912 \end{bmatrix}$	$\mathbb{D}$ $[0.41]$ -	$\Delta[\Xi_u]$ 0.2034 $\Delta[\mathbb{D}_*]^{\ominus}$ 0.2978
I_2	$\begin{bmatrix} -0.579 & -0.597 & 0.555 \\ 0.815 & -0.440 & 0.377 \\ 0.019 & 0.671 & 0.741 \end{bmatrix}$	$\mathbb{D}$ $[0.67]$ -	$\Delta[\Xi_u]$ 0.1472 $\Delta[\mathbb{D}_*]^{\ominus}$ 0.0360
I_3	$\begin{bmatrix} -0.875 & 0.301 & -0.379 \\ 0.377 & 0.915 & -0.144 \\ 0.304 & -0.269 & -0.914 \end{bmatrix}$	$\mathbb{D}$ $[0.41]$ -	$\Delta[\Xi_u]$ 0.1924 $\Delta[\mathbb{D}_*]^{\ominus}$ 0.3014
I_4	$\begin{bmatrix} -0.309 & 0.705 & -0.638 \\ 0.949 & 0.184 & -0.256 \\ -0.063 & -0.684 & -0.726 \end{bmatrix}$	$\mathbb{D}$ $[0.69]$ -	$\Delta[\Xi_u]$ 0.1492 $\Delta[\mathbb{D}_*]^{\ominus}$ 0.0198

(c) Hypothesis H_1 for $n_\theta = 10$.

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\mathbb{D}_*]^{\ominus}$
H_1	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 0.707 & -0.707 \\ 0.000 & 0.707 & 0.707 \end{bmatrix}$	$\mathbb{D}$ $[0.71]$ -	
I_0	$\begin{bmatrix} 0.855 & 0.343 & 0.389 \\ 0.518 & -0.601 & -0.608 \\ 0.026 & 0.722 & -0.692 \end{bmatrix}$	$\mathbb{D}$ $[0.72]$ -	$\Delta[\Xi_u]$ 0.1622 $\Delta[\mathbb{D}_*]^{\ominus}$ 0.0152
I_1	$\begin{bmatrix} -0.755 & 0.475 & -0.453 \\ -0.656 & -0.526 & 0.542 \\ 0.019 & 0.706 & 0.708 \end{bmatrix}$	$\mathbb{D}$ $[0.71]$ -	$\Delta[\Xi_u]$ 0.1599 $\Delta[\mathbb{D}_*]^{\ominus}$ 0.0011
I_2	$\begin{bmatrix} -0.991 & 0.084 & -0.102 \\ -0.132 & -0.685 & 0.717 \\ -0.010 & 0.724 & 0.690 \end{bmatrix}$	$\mathbb{D}$ $[0.72]$ -	$\Delta[\Xi_u]$ 0.1481 $\Delta[\mathbb{D}_*]^{\ominus}$ 0.0170
I_3	$\begin{bmatrix} 0.837 & -0.358 & -0.415 \\ 0.546 & 0.597 & 0.588 \\ 0.037 & -0.718 & 0.695 \end{bmatrix}$	$\mathbb{D}$ $[0.72]$ -	$\Delta[\Xi_u]$ 0.1531 $\Delta[\mathbb{D}_*]^{\ominus}$ 0.0123
I_4	$\begin{bmatrix} -0.188 & 0.721 & -0.668 \\ 0.979 & 0.086 & -0.183 \\ -0.075 & -0.688 & -0.722 \end{bmatrix}$	$\mathbb{D}$ $[0.69]$ -	$\Delta[\Xi_u]$ 0.1459 $\Delta[\mathbb{D}_*]^{\ominus}$ 0.0149

(d) Hypothesis H_1 for $n_\theta = 20$.

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\mathbb{D}_*]^{\ominus}$
H_2	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 0.383 & -0.924 \\ 0.000 & 0.924 & 0.383 \end{bmatrix}$	$\mathbb{D}$ $[0.92]$ -	
I_0	$\begin{bmatrix} 0.185 & -0.392 & 0.901 \\ 0.983 & 0.057 & -0.177 \\ 0.018 & 0.918 & 0.396 \end{bmatrix}$	$\mathbb{D}$ $[0.92]$ -	$\Delta[\Xi_u]$ 0.1520 $\Delta[\mathbb{D}_*]^{\ominus}$ 0.0054
I_1	$\begin{bmatrix} 0.695 & 0.266 & -0.668 \\ -0.719 & 0.272 & -0.640 \\ 0.011 & 0.925 & 0.380 \end{bmatrix}$	$\mathbb{D}$ $[0.92]$ -	$\Delta[\Xi_u]$ 0.1645 $\Delta[\mathbb{D}_*]^{\ominus}$ 0.0010
I_2	$\begin{bmatrix} -0.168 & -0.344 & 0.924 \\ 0.985 & -0.086 & 0.147 \\ 0.029 & 0.935 & 0.353 \end{bmatrix}$	$\mathbb{D}$ $[0.94]$ -	$\Delta[\Xi_u]$ 0.1587 $\Delta[\mathbb{D}_*]^{\ominus}$ 0.0118
I_3	$\begin{bmatrix} -0.162 & -0.348 & 0.924 \\ -0.967 & 0.243 & -0.078 \\ -0.197 & -0.906 & -0.375 \end{bmatrix}$	$\mathbb{D}$ $[0.93]$ -	$\Delta[\Xi_u]$ 0.1205 $\Delta[\mathbb{D}_*]^{\ominus}$ 0.0030
I_4	$\begin{bmatrix} 0.434 & 0.432 & -0.790 \\ 0.901 & -0.184 & 0.394 \\ 0.025 & -0.883 & -0.469 \end{bmatrix}$	$\mathbb{D}$ $[0.88]$ -	$\Delta[\Xi_u]$ 0.1575 $\Delta[\mathbb{D}_*]^{\ominus}$ 0.0407

(e) Hypothesis H_2 for $n_\theta = 10$.

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\mathbb{D}_*]^{\ominus}$
H_2	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 0.383 & -0.924 \\ 0.000 & 0.924 & 0.383 \end{bmatrix}$	$\mathbb{D}$ $[0.92]$ -	
I_0	$\begin{bmatrix} 0.148 & -0.394 & -0.907 \\ -0.989 & -0.043 & -0.142 \\ 0.017 & 0.918 & -0.396 \end{bmatrix}$	$\mathbb{D}$ $[0.92]$ -	$\Delta[\Xi_u]$ 0.1674 $\Delta[\mathbb{D}_*]^{\ominus}$ 0.0056
I_1	$\begin{bmatrix} -0.155 & -0.388 & 0.908 \\ 0.987 & -0.097 & 0.127 \\ 0.038 & 0.916 & 0.398 \end{bmatrix}$	$\mathbb{D}$ $[0.92]$ -	$\Delta[\Xi_u]$ 0.1609 $\Delta[\mathbb{D}_*]^{\ominus}$ 0.0066
I_2	$\begin{bmatrix} 0.996 & 0.021 & -0.082 \\ -0.084 & 0.368 & -0.926 \\ 0.010 & 0.930 & 0.368 \end{bmatrix}$	$\mathbb{D}$ $[0.93]$ -	$\Delta[\Xi_u]$ 0.1606 $\Delta[\mathbb{D}_*]^{\ominus}$ 0.0059
I_3	$\begin{bmatrix} 0.692 & -0.384 & -0.611 \\ 0.671 & 0.032 & 0.741 \\ -0.265 & -0.923 & 0.280 \end{bmatrix}$	$\mathbb{D}$ $[0.96]$ -	$\Delta[\Xi_u]$ 0.1260 $\Delta[\mathbb{D}_*]^{\ominus}$ 0.0360
I_4	$\begin{bmatrix} 0.936 & 0.145 & 0.320 \\ -0.352 & 0.389 & 0.852 \\ -0.001 & -0.910 & 0.415 \end{bmatrix}$	$\mathbb{D}$ $[0.91]$ -	$\Delta[\Xi_u]$ 0.1534 $\Delta[\mathbb{D}_*]^{\ominus}$ 0.0140

(f) Hypothesis H_2 for $n_\theta = 20$.

Table D.67: Noisy optimization parameterized with SO(n) on $\mathrm{H}_3^{\oplus}\mathrm{SE}(2)$. Sensor and actuator noise set to $\sigma_c, \sigma_p = 0.05$. Hypotheses H_0 to H_2. True projection matrix and optimization results per iteration I_0 to I_4. Commutativity vector, unnormalized intrinsic conflict and commutativity delta.

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\oplus_*]^{\ominus}$
H_3	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 1.000 & 0.000 \\ 0.000 & 0.000 & 1.000 \end{bmatrix}$	ⓘ [0.00]	
I_0	$\begin{bmatrix} 0.620 & 0.784 & -0.011 \\ 0.784 & -0.620 & -0.001 \\ -0.008 & -0.008 & -1.000 \end{bmatrix}$	ⓘ [0.01]	$\Delta[\Xi_u]$ 0.1218 $\Delta[\oplus_*]^{\ominus}$ 0.0113
I_1	$\begin{bmatrix} -0.974 & -0.035 & 0.226 \\ -0.081 & 0.977 & -0.195 \\ -0.214 & -0.208 & -0.955 \end{bmatrix}$	ⓘ [0.30]	$\Delta[\Xi_u]$ 0.1697 $\Delta[\oplus_*]^{\ominus}$ 0.2982
I_2	$\begin{bmatrix} -0.069 & 0.998 & 0.003 \\ 0.998 & 0.068 & 0.016 \\ 0.016 & 0.004 & -1.000 \end{bmatrix}$	ⓘ [0.02]	$\Delta[\Xi_u]$ 0.0959 $\Delta[\oplus_*]^{\ominus}$ 0.0164
I_3	$\begin{bmatrix} -0.954 & -0.273 & -0.127 \\ 0.237 & -0.940 & 0.247 \\ -0.187 & 0.205 & 0.961 \end{bmatrix}$	ⓘ [0.28]	$\Delta[\Xi_u]$ 0.1424 $\Delta[\oplus_*]^{\ominus}$ 0.2774
I_4	$\begin{bmatrix} -0.184 & 0.983 & -0.012 \\ 0.983 & 0.184 & -0.004 \\ -0.002 & -0.013 & -1.000 \end{bmatrix}$	ⓘ [0.01]	$\Delta[\Xi_u]$ 0.0912 $\Delta[\oplus_*]^{\ominus}$ 0.0131

(a) Hypothesis H_3 for $n_\theta = 10$.

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\oplus_*]^{\ominus}$
H_3	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 1.000 & 0.000 \\ 0.000 & 0.000 & 1.000 \end{bmatrix}$	ⓘ [0.00]	
I_0	$\begin{bmatrix} 0.640 & 0.768 & -0.021 \\ 0.768 & -0.640 & -0.012 \\ -0.023 & -0.009 & -1.000 \end{bmatrix}$	ⓘ [0.02]	$\Delta[\Xi_u]$ 0.1198 $\Delta[\oplus_*]^{\ominus}$ 0.0243
I_1	$\begin{bmatrix} 0.992 & 0.128 & -0.001 \\ -0.128 & 0.992 & 0.004 \\ 0.001 & -0.004 & 1.000 \end{bmatrix}$	ⓘ [0.00]	$\Delta[\Xi_u]$ 0.0842 $\Delta[\oplus_*]^{\ominus}$ 0.0040
I_2	$\begin{bmatrix} -0.120 & 0.993 & -0.003 \\ 0.993 & 0.120 & 0.009 \\ 0.009 & -0.002 & -1.000 \end{bmatrix}$	ⓘ [0.01]	$\Delta[\Xi_u]$ 0.0936 $\Delta[\oplus_*]^{\ominus}$ 0.0095
I_3	$\begin{bmatrix} -0.697 & -0.717 & -0.003 \\ 0.717 & -0.697 & 0.001 \\ -0.003 & -0.002 & 1.000 \end{bmatrix}$	ⓘ [0.00]	$\Delta[\Xi_u]$ 0.0760 $\Delta[\oplus_*]^{\ominus}$ 0.0036
I_4	$\begin{bmatrix} 0.967 & 0.254 & 0.000 \\ -0.254 & 0.967 & -0.011 \\ -0.003 & 0.011 & 1.000 \end{bmatrix}$	ⓘ [0.01]	$\Delta[\Xi_u]$ 0.0864 $\Delta[\oplus_*]^{\ominus}$ 0.0114

(b) Hypothesis H_3 for $n_\theta = 20$.

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\oplus_*]^{\ominus}$
H_4	$\begin{bmatrix} 0.000 & 0.000 & 1.000 \\ 1.000 & 0.000 & 0.000 \\ 0.000 & 1.000 & 0.000 \end{bmatrix}$	ⓘ [1.00]	
I_0	$\begin{bmatrix} -0.900 & 0.064 & 0.430 \\ 0.432 & 0.022 & 0.901 \\ 0.048 & 0.998 & -0.047 \end{bmatrix}$	ⓘ [1.00]	$\Delta[\Xi_u]$ 0.1314 $\Delta[\oplus_*]^{\ominus}$ 0.0011
I_1	$\begin{bmatrix} 0.207 & 0.051 & 0.977 \\ 0.978 & -0.034 & -0.206 \\ 0.023 & 0.998 & -0.057 \end{bmatrix}$	ⓘ [1.00]	$\Delta[\Xi_u]$ 0.1370 $\Delta[\oplus_*]^{\ominus}$ 0.0016
I_2	$\begin{bmatrix} -0.537 & 0.026 & 0.843 \\ 0.843 & 0.013 & 0.537 \\ 0.003 & 1.000 & -0.029 \end{bmatrix}$	ⓘ [1.00]	$\Delta[\Xi_u]$ 0.1705 $\Delta[\oplus_*]^{\ominus}$ 0.0004
I_3	$\begin{bmatrix} -0.995 & -0.029 & -0.097 \\ -0.098 & 0.015 & 0.995 \\ -0.028 & 0.999 & -0.018 \end{bmatrix}$	ⓘ [1.00]	$\Delta[\Xi_u]$ 0.1276 $\Delta[\oplus_*]^{\ominus}$ 0.0002
I_4	$\begin{bmatrix} 0.919 & -0.061 & -0.390 \\ 0.394 & 0.192 & 0.899 \\ 0.020 & -0.980 & 0.200 \end{bmatrix}$	ⓘ [0.98]	$\Delta[\Xi_u]$ 0.1480 $\Delta[\oplus_*]^{\ominus}$ 0.0202

(c) Hypothesis H_4 for $n_\theta = 10$.

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\oplus_*]^{\ominus}$
H_4	$\begin{bmatrix} 0.000 & 0.000 & 1.000 \\ 1.000 & 0.000 & 0.000 \\ 0.000 & 1.000 & 0.000 \end{bmatrix}$	ⓘ [1.00]	
I_0	$\begin{bmatrix} 0.272 & -0.081 & -0.959 \\ 0.961 & -0.020 & 0.274 \\ -0.042 & -0.997 & 0.072 \end{bmatrix}$	ⓘ [1.00]	$\Delta[\Xi_u]$ 0.1307 $\Delta[\oplus_*]^{\ominus}$ 0.0026
I_1	$\begin{bmatrix} 0.955 & -0.011 & 0.296 \\ -0.294 & 0.081 & 0.952 \\ -0.034 & -0.997 & 0.075 \end{bmatrix}$	ⓘ [1.00]	$\Delta[\Xi_u]$ 0.1348 $\Delta[\oplus_*]^{\ominus}$ 0.0028
I_2	$\begin{bmatrix} -0.705 & 0.004 & 0.710 \\ 0.709 & 0.018 & 0.705 \\ -0.010 & 1.000 & -0.015 \end{bmatrix}$	ⓘ [1.00]	$\Delta[\Xi_u]$ 0.1712 $\Delta[\oplus_*]^{\ominus}$ 0.0001
I_3	$\begin{bmatrix} -0.399 & -0.023 & 0.917 \\ -0.915 & -0.056 & -0.400 \\ 0.060 & -0.998 & 0.001 \end{bmatrix}$	ⓘ [1.00]	$\Delta[\Xi_u]$ 0.1287 $\Delta[\oplus_*]^{\ominus}$ 0.0000
I_4	$\begin{bmatrix} -0.895 & 0.027 & 0.445 \\ 0.443 & 0.166 & 0.881 \\ -0.050 & 0.986 & -0.161 \end{bmatrix}$	ⓘ [0.99]	$\Delta[\Xi_u]$ 0.1456 $\Delta[\oplus_*]^{\ominus}$ 0.0130

(d) Hypothesis H_4 for $n_\theta = 20$.

Table D.68: Noisy optimization parameterized with SO(n) on $\mathrm{H}_3^{\oplus}\mathrm{SE}(2)$. Sensor and actuator noise set to $\sigma_c, \sigma_p = 0.05$. Hypotheses H_3 to H_4. True projection matrix and optimization results per iteration I_0 to I_4. Commutativity vector, unnormalized intrinsic conflict and commutativity delta.

Id	Hypothesis, $\Delta[\oplus_*]^{\ominus}$				
	H_0	H_1	H_2	H_3	H_4
I_0	0.0039	0.0311	0.0054	0.0113	0.0011
I_1	0.0019	0.2978	0.0010	0.2982	0.0016
I_2	0.0024	0.0360	0.0118	0.0164	0.0004
I_3	0.0254	0.3014	0.0030	0.2774	0.0002
I_4	0.0021	0.0198	0.0407	0.0131	0.0202
μ	0.0071	0.1372	0.0124	0.1233	0.0047
σ	0.0092	0.1327	0.0146	0.1345	0.0078
$\mu = 0.0569$, $\sigma = 0.1062$					

(a) Evaluation for $n_\theta = 10$.

Id	Hypothesis, $\Delta[\oplus_*]^{\ominus}$				
	H_0	H_1	H_2	H_3	H_4
I_0	0.0107	0.0152	0.0056	0.0243	0.0026
I_1	0.0035	0.0011	0.0066	0.0040	0.0028
I_2	0.0067	0.0170	0.0059	0.0095	0.0001
I_3	0.0207	0.0123	0.0360	0.0036	0.0000
I_4	0.0094	0.0149	0.0140	0.0114	0.0130
μ	0.0102	0.0121	0.0136	0.0106	0.0037
σ	0.0058	0.0057	0.0116	0.0075	0.0048
$\mu = 0.0100$, $\sigma = 0.0084$					

(b) Evaluation for $n_\theta = 20$.

Table D.69: Noisy optimization parameterized with SO(n) on $\mathrm{H}_3^{\ominus}\mathrm{SE}(2)$. Sensor and actuator noise set to $\sigma_c, \sigma_p = 0.05$. Commutativity delta $\Delta[\oplus_*]^{\ominus}$ per iteration I_0 to I_4. Mean and standard deviation of $\Delta[\oplus_*]^{\ominus}$ per hypothesis H_0 to H_4 and over all hypotheses.

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\oplus_*]^{\ominus}$
H_0	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 0.924 & -0.383 \\ 0.000 & 0.383 & 0.924 \end{bmatrix}$	⊕ [0.38] -	
I_0	$\begin{bmatrix} 0.453 & 0.816 & -0.360 \\ 0.891 & -0.424 & 0.161 \\ -0.021 & -0.394 & -0.919 \end{bmatrix}$	⊕ [0.39] -	$\Delta[\Xi_u]$ 0.2004 $\Delta[\oplus_*]^{\ominus}$ 0.0116
I_1	$\begin{bmatrix} -0.948 & -0.279 & 0.153 \\ -0.314 & 0.899 & -0.305 \\ -0.052 & -0.337 & -0.940 \end{bmatrix}$	⊕ [0.34] -	$\Delta[\Xi_u]$ 0.1946 $\Delta[\oplus_*]^{\ominus}$ 0.0413
I_2	$\begin{bmatrix} -0.780 & -0.584 & 0.226 \\ 0.626 & -0.736 & 0.258 \\ 0.016 & 0.343 & 0.939 \end{bmatrix}$	⊕ [0.34] -	$\Delta[\Xi_u]$ 0.2280 $\Delta[\oplus_*]^{\ominus}$ 0.0395
I_3	$\begin{bmatrix} -0.781 & 0.606 & -0.149 \\ 0.621 & 0.727 & -0.293 \\ -0.069 & -0.322 & -0.944 \end{bmatrix}$	⊕ [0.33] -	$\Delta[\Xi_u]$ 0.1731 $\Delta[\oplus_*]^{\ominus}$ 0.0537
I_4	$\begin{bmatrix} 0.289 & 0.861 & -0.420 \\ -0.956 & 0.283 & -0.078 \\ 0.052 & 0.424 & 0.904 \end{bmatrix}$	⊕ [0.43] -	$\Delta[\Xi_u]$ 0.1970 $\Delta[\oplus_*]^{\ominus}$ 0.0440

(a) Hypothesis H_0 for $n_\theta = 10$.

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\oplus_*]^{\ominus}$
H_0	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 0.924 & -0.383 \\ 0.000 & 0.383 & 0.924 \end{bmatrix}$	⊕ [0.38] -	
I_0	$\begin{bmatrix} 0.602 & 0.733 & -0.316 \\ 0.799 & -0.556 & 0.232 \\ -0.006 & -0.392 & -0.920 \end{bmatrix}$	⊕ [0.39] -	$\Delta[\Xi_u]$ 0.2013 $\Delta[\oplus_*]^{\ominus}$ 0.0094
I_1	$\begin{bmatrix} -0.964 & -0.234 & 0.122 \\ -0.262 & 0.910 & -0.321 \\ -0.036 & -0.342 & -0.939 \end{bmatrix}$	⊕ [0.34] -	$\Delta[\Xi_u]$ 0.1939 $\Delta[\oplus_*]^{\ominus}$ 0.0391
I_2	$\begin{bmatrix} -0.936 & -0.328 & 0.129 \\ -0.352 & 0.869 & -0.348 \\ 0.002 & -0.371 & -0.929 \end{bmatrix}$	⊕ [0.37] -	$\Delta[\Xi_u]$ 0.2284 $\Delta[\oplus_*]^{\ominus}$ 0.0117
I_3	$\begin{bmatrix} -0.593 & 0.785 & -0.181 \\ 0.798 & 0.543 & -0.261 \\ -0.107 & -0.299 & -0.948 \end{bmatrix}$	⊕ [0.32] -	$\Delta[\Xi_u]$ 0.1757 $\Delta[\oplus_*]^{\ominus}$ 0.0650
I_4	$\begin{bmatrix} 0.940 & 0.293 & -0.175 \\ -0.332 & 0.900 & -0.280 \\ 0.075 & 0.322 & 0.944 \end{bmatrix}$	⊕ [0.33] -	$\Delta[\Xi_u]$ 0.1955 $\Delta[\oplus_*]^{\ominus}$ 0.0523

(b) Hypothesis H_0 for $n_\theta = 20$.

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\oplus_*]^{\ominus}$
H_1	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 0.707 & -0.707 \\ 0.000 & 0.707 & 0.707 \end{bmatrix}$	⊕ [0.71] -	
I_0	$\begin{bmatrix} 0.670 & 0.484 & 0.563 \\ 0.741 & -0.473 & -0.476 \\ 0.036 & 0.736 & -0.676 \end{bmatrix}$	⊕ [0.74] -	$\Delta[\Xi_u]$ 0.2077 $\Delta[\oplus_*]^{\ominus}$ 0.0299
I_1	$\begin{bmatrix} -0.735 & 0.653 & -0.181 \\ 0.616 & 0.533 & -0.580 \\ -0.283 & -0.538 & -0.794 \end{bmatrix}$	⊕ [0.61] -	$\Delta[\Xi_u]$ 0.2513 $\Delta[\oplus_*]^{\ominus}$ 0.0994
I_2	$\begin{bmatrix} -0.616 & -0.561 & 0.553 \\ 0.788 & -0.439 & 0.432 \\ -0.000 & 0.702 & 0.712 \end{bmatrix}$	⊕ [0.70] -	$\Delta[\Xi_u]$ 0.2308 $\Delta[\oplus_*]^{\ominus}$ 0.0053
I_3	$\begin{bmatrix} -0.434 & -0.635 & 0.639 \\ -0.901 & 0.280 & -0.333 \\ 0.032 & -0.720 & -0.693 \end{bmatrix}$	⊕ [0.72] -	$\Delta[\Xi_u]$ 0.2193 $\Delta[\oplus_*]^{\ominus}$ 0.0137
I_4	$\begin{bmatrix} 0.433 & -0.658 & 0.617 \\ 0.827 & 0.562 & 0.018 \\ -0.358 & 0.502 & 0.787 \end{bmatrix}$	⊕ [0.62] -	$\Delta[\Xi_u]$ 0.2246 $\Delta[\oplus_*]^{\ominus}$ 0.0902

(c) Hypothesis H_1 for $n_\theta = 10$.

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\oplus_*]^{\ominus}$
H_1	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 0.707 & -0.707 \\ 0.000 & 0.707 & 0.707 \end{bmatrix}$	⊕ [0.71] -	
I_0	$\begin{bmatrix} -0.809 & -0.398 & 0.432 \\ -0.587 & 0.557 & -0.587 \\ -0.007 & -0.729 & -0.685 \end{bmatrix}$	⊕ [0.73] -	$\Delta[\Xi_u]$ 0.1817 $\Delta[\oplus_*]^{\ominus}$ 0.0219
I_1	$\begin{bmatrix} 0.188 & 0.710 & -0.679 \\ -0.982 & 0.135 & -0.131 \\ -0.002 & 0.692 & 0.722 \end{bmatrix}$	⊕ [0.69] -	$\Delta[\Xi_u]$ 0.2284 $\Delta[\oplus_*]^{\ominus}$ 0.0155
I_2	$\begin{bmatrix} -0.881 & -0.345 & 0.322 \\ 0.472 & -0.624 & 0.623 \\ -0.014 & 0.701 & 0.713 \end{bmatrix}$	⊕ [0.70] -	$\Delta[\Xi_u]$ 0.2303 $\Delta[\oplus_*]^{\ominus}$ 0.0059
I_3	$\begin{bmatrix} -0.169 & -0.638 & 0.752 \\ -0.984 & 0.153 & -0.091 \\ -0.057 & -0.755 & -0.653 \end{bmatrix}$	⊕ [0.76] -	$\Delta[\Xi_u]$ 0.2207 $\Delta[\oplus_*]^{\ominus}$ 0.0500
I_4	$\begin{bmatrix} -0.237 & 0.768 & -0.594 \\ 0.917 & 0.380 & 0.125 \\ 0.322 & -0.515 & -0.794 \end{bmatrix}$	⊕ [0.61] -	$\Delta[\Xi_u]$ 0.2216 $\Delta[\oplus_*]^{\ominus}$ 0.0998

(d) Hypothesis H_1 for $n_\theta = 20$.

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\oplus_*]^{\ominus}$
H_2	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 0.383 & -0.924 \\ 0.000 & 0.924 & 0.383 \end{bmatrix}$	⊕ [0.92] -	
I_0	$\begin{bmatrix} -0.570 & -0.372 & 0.732 \\ 0.820 & -0.309 & 0.481 \\ 0.047 & 0.875 & 0.482 \end{bmatrix}$	⊕ [0.88] -	$\Delta[\Xi_u]$ 0.2328 $\Delta[\oplus_*]^{\ominus}$ 0.0475
I_1	$\begin{bmatrix} 0.702 & -0.354 & -0.618 \\ -0.710 & -0.277 & -0.647 \\ 0.057 & 0.893 & -0.446 \end{bmatrix}$	⊕ [0.90] -	$\Delta[\Xi_u]$ 0.2166 $\Delta[\oplus_*]^{\ominus}$ 0.0287
I_2	$\begin{bmatrix} 0.712 & -0.303 & -0.634 \\ -0.702 & -0.270 & -0.659 \\ 0.029 & 0.914 & -0.404 \end{bmatrix}$	⊕ [0.91] -	$\Delta[\Xi_u]$ 0.2332 $\Delta[\oplus_*]^{\ominus}$ 0.0093
I_3	$\begin{bmatrix} -0.991 & -0.081 & 0.109 \\ 0.089 & 0.220 & 0.972 \\ -0.102 & 0.972 & -0.211 \end{bmatrix}$	⊕ [0.98] -	$\Delta[\Xi_u]$ 0.2139 $\Delta[\oplus_*]^{\ominus}$ 0.0537
I_4	$\begin{bmatrix} 0.033 & 0.469 & -0.883 \\ 0.990 & -0.136 & -0.035 \\ -0.136 & -0.873 & -0.469 \end{bmatrix}$	⊕ [0.88] -	$\Delta[\Xi_u]$ 0.2474 $\Delta[\oplus_*]^{\ominus}$ 0.0405

(e) Hypothesis H_2 for $n_\theta = 10$.

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\oplus_*]^{\ominus}$
H_2	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 0.383 & -0.924 \\ 0.000 & 0.924 & 0.383 \end{bmatrix}$	⊕ [0.92] -	
I_0	$\begin{bmatrix} -0.965 & -0.052 & 0.255 \\ -0.211 & 0.729 & -0.651 \\ -0.152 & -0.682 & -0.715 \end{bmatrix}$	⊕ [0.70] -	$\Delta[\Xi_u]$ 0.2387 $\Delta[\oplus_*]^{\ominus}$ 0.2249
I_1	$\begin{bmatrix} 0.954 & 0.081 & -0.289 \\ -0.294 & 0.447 & -0.845 \\ 0.061 & 0.891 & 0.451 \end{bmatrix}$	⊕ [0.89] -	$\Delta[\Xi_u]$ 0.2111 $\Delta[\oplus_*]^{\ominus}$ 0.0311
I_2	$\begin{bmatrix} -0.990 & -0.015 & 0.141 \\ 0.136 & -0.369 & 0.920 \\ 0.038 & 0.929 & 0.367 \end{bmatrix}$	⊕ [0.93] -	$\Delta[\Xi_u]$ 0.2215 $\Delta[\oplus_*]^{\ominus}$ 0.0064
I_3	$\begin{bmatrix} -0.396 & -0.425 & 0.814 \\ -0.916 & 0.118 & -0.383 \\ 0.067 & -0.897 & -0.436 \end{bmatrix}$	⊕ [0.90] -	$\Delta[\Xi_u]$ 0.2068 $\Delta[\oplus_*]^{\ominus}$ 0.0240
I_4	$\begin{bmatrix} -0.036 & 0.436 & -0.899 \\ 0.990 & -0.104 & -0.091 \\ -0.133 & -0.894 & -0.428 \end{bmatrix}$	⊕ [0.90] -	$\Delta[\Xi_u]$ 0.2483 $\Delta[\oplus_*]^{\ominus}$ 0.0200

(f) Hypothesis H_2 for $n_\theta = 20$.

Table D.70: Noisy optimization parameterized with SO(n) on $\mathrm{H}_3^{\oplus}\mathrm{SE}(2)$. Sensor and actuator noise set to $\sigma_c, \sigma_p = 0.10$. Hypotheses H_0 to H_2. True projection matrix and optimization results per iteration I_0 to I_4. Commutativity vector, unnormalized intrinsic conflict and commutativity delta.

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\mathbb{O}_*]^{\ominus}$
H_3	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 1.000 & 0.000 \\ 0.000 & 0.000 & 1.000 \end{bmatrix}$	$\mathbb{O}$ [0.00] -	
I_0	$\begin{bmatrix} 0.617 & 0.785 & -0.050 \\ 0.786 & -0.619 & -0.008 \\ -0.037 & -0.034 & -0.999 \end{bmatrix}$	$\mathbb{O}$ [0.05] -	$\Delta[\Xi_u]$ 0.2192 $\Delta[\mathbb{O}_*]^{\ominus}$ 0.0507
I_1	$\begin{bmatrix} -0.147 & -0.989 & 0.013 \\ -0.989 & 0.147 & -0.007 \\ 0.005 & -0.014 & -1.000 \end{bmatrix}$	$\mathbb{O}$ [0.01] -	$\Delta[\Xi_u]$ 0.1622 $\Delta[\mathbb{O}_*]^{\ominus}$ 0.0147
I_2	$\begin{bmatrix} -0.269 & 0.963 & 0.004 \\ 0.963 & 0.269 & 0.012 \\ 0.011 & 0.007 & -1.000 \end{bmatrix}$	$\mathbb{O}$ [0.01] -	$\Delta[\Xi_u]$ 0.1550 $\Delta[\mathbb{O}_*]^{\ominus}$ 0.0127
I_3	$\begin{bmatrix} -0.823 & 0.497 & -0.274 \\ 0.533 & 0.844 & -0.070 \\ 0.197 & -0.203 & -0.959 \end{bmatrix}$	$\mathbb{O}$ [0.28] -	$\Delta[\Xi_u]$ 0.1914 $\Delta[\mathbb{O}_*]^{\ominus}$ 0.2829
I_4	$\begin{bmatrix} 0.988 & -0.155 & 0.015 \\ 0.156 & 0.987 & -0.047 \\ -0.008 & 0.049 & 0.999 \end{bmatrix}$	$\mathbb{O}$ [0.05] -	$\Delta[\Xi_u]$ 0.1603 $\Delta[\mathbb{O}_*]^{\ominus}$ 0.0491

(a) Hypothesis H_3 for $n_\theta = 10$.

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\mathbb{O}_*]^{\ominus}$
H_3	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 1.000 & 0.000 \\ 0.000 & 0.000 & 1.000 \end{bmatrix}$	$\mathbb{O}$ [0.00] -	
I_0	$\begin{bmatrix} 0.614 & 0.789 & -0.038 \\ 0.789 & -0.614 & -0.009 \\ -0.031 & -0.024 & -0.999 \end{bmatrix}$	$\mathbb{O}$ [0.04] -	$\Delta[\Xi_u]$ 0.2187 $\Delta[\mathbb{O}_*]^{\ominus}$ 0.0390
I_1	$\begin{bmatrix} 0.119 & -0.993 & 0.007 \\ -0.993 & -0.119 & 0.007 \\ -0.006 & -0.008 & -1.000 \end{bmatrix}$	$\mathbb{O}$ [0.01] -	$\Delta[\Xi_u]$ 0.1727 $\Delta[\mathbb{O}_*]^{\ominus}$ 0.0103
I_2	$\begin{bmatrix} -0.756 & 0.654 & -0.027 \\ 0.654 & 0.756 & 0.017 \\ 0.032 & -0.005 & -0.999 \end{bmatrix}$	$\mathbb{O}$ [0.03] -	$\Delta[\Xi_u]$ 0.1688 $\Delta[\mathbb{O}_*]^{\ominus}$ 0.0321
I_3	$\begin{bmatrix} -0.930 & -0.131 & 0.344 \\ -0.073 & 0.981 & 0.177 \\ -0.361 & 0.139 & -0.922 \end{bmatrix}$	$\mathbb{O}$ [0.39] -	$\Delta[\Xi_u]$ 0.1827 $\Delta[\mathbb{O}_*]^{\ominus}$ 0.3870
I_4	$\begin{bmatrix} 0.971 & 0.099 & -0.216 \\ 0.072 & -0.989 & -0.131 \\ -0.226 & 0.111 & -0.968 \end{bmatrix}$	$\mathbb{O}$ [0.25] -	$\Delta[\Xi_u]$ 0.1873 $\Delta[\mathbb{O}_*]^{\ominus}$ 0.2524

(b) Hypothesis H_3 for $n_\theta = 20$.

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\mathbb{O}_*]^{\ominus}$
H_4	$\begin{bmatrix} 0.000 & 0.000 & 1.000 \\ 1.000 & 0.000 & 0.000 \\ 0.000 & 1.000 & 0.000 \end{bmatrix}$	$\mathbb{O}$ [1.00] -	
I_0	$\begin{bmatrix} 0.999 & -0.023 & -0.045 \\ 0.040 & -0.193 & 0.980 \\ -0.031 & -0.981 & -0.192 \end{bmatrix}$	$\mathbb{O}$ [0.98] -	$\Delta[\Xi_u]$ 0.1857 $\Delta[\mathbb{O}_*]^{\ominus}$ 0.0185
I_1	$\begin{bmatrix} 0.193 & 0.228 & 0.954 \\ 0.981 & -0.079 & -0.179 \\ 0.035 & 0.970 & -0.239 \end{bmatrix}$	$\mathbb{O}$ [0.97] -	$\Delta[\Xi_u]$ 0.2390 $\Delta[\mathbb{O}_*]^{\ominus}$ 0.0291
I_2	$\begin{bmatrix} 0.999 & -0.034 & 0.015 \\ 0.016 & 0.017 & -1.000 \\ 0.034 & 0.999 & 0.018 \end{bmatrix}$	$\mathbb{O}$ [1.00] -	$\Delta[\Xi_u]$ 0.2354 $\Delta[\mathbb{O}_*]^{\ominus}$ 0.0002
I_3	$\begin{bmatrix} -0.981 & -0.083 & -0.176 \\ -0.176 & -0.009 & 0.984 \\ -0.083 & 0.997 & -0.006 \end{bmatrix}$	$\mathbb{O}$ [1.00] -	$\Delta[\Xi_u]$ 0.2258 $\Delta[\mathbb{O}_*]^{\ominus}$ 0.0000
I_4	$\begin{bmatrix} 0.749 & -0.076 & -0.658 \\ 0.661 & 0.018 & 0.750 \\ -0.045 & -0.997 & 0.064 \end{bmatrix}$	$\mathbb{O}$ [1.00] -	$\Delta[\Xi_u]$ 0.2524 $\Delta[\mathbb{O}_*]^{\ominus}$ 0.0020

(c) Hypothesis H_4 for $n_\theta = 10$.

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\mathbb{O}_*]^{\ominus}$
H_4	$\begin{bmatrix} 0.000 & 0.000 & 1.000 \\ 1.000 & 0.000 & 0.000 \\ 0.000 & 1.000 & 0.000 \end{bmatrix}$	$\mathbb{O}$ [1.00] -	
I_0	$\begin{bmatrix} 0.344 & 0.191 & -0.919 \\ 0.938 & -0.106 & 0.330 \\ -0.034 & -0.976 & -0.215 \end{bmatrix}$	$\mathbb{O}$ [0.98] -	$\Delta[\Xi_u]$ 0.1845 $\Delta[\mathbb{O}_*]^{\ominus}$ 0.0235
I_1	$\begin{bmatrix} 0.707 & -0.180 & -0.684 \\ -0.707 & -0.135 & -0.694 \\ 0.033 & 0.974 & -0.223 \end{bmatrix}$	$\mathbb{O}$ [0.97] -	$\Delta[\Xi_u]$ 0.2390 $\Delta[\mathbb{O}_*]^{\ominus}$ 0.0252
I_2	$\begin{bmatrix} 0.426 & 0.048 & -0.903 \\ 0.904 & 0.014 & 0.427 \\ 0.034 & -0.999 & -0.037 \end{bmatrix}$	$\mathbb{O}$ [1.00] -	$\Delta[\Xi_u]$ 0.2403 $\Delta[\mathbb{O}_*]^{\ominus}$ 0.0007
I_3	$\begin{bmatrix} -0.348 & -0.030 & 0.937 \\ -0.934 & -0.079 & -0.349 \\ 0.084 & -0.996 & -0.000 \end{bmatrix}$	$\mathbb{O}$ [1.00] -	$\Delta[\Xi_u]$ 0.2270 $\Delta[\mathbb{O}_*]^{\ominus}$ 0.0000
I_4	$\begin{bmatrix} -0.525 & 0.253 & -0.812 \\ 0.850 & 0.127 & -0.510 \\ -0.026 & -0.959 & -0.282 \end{bmatrix}$	$\mathbb{O}$ [0.96] -	$\Delta[\Xi_u]$ 0.2510 $\Delta[\mathbb{O}_*]^{\ominus}$ 0.0406

(d) Hypothesis H_4 for $n_\theta = 20$.

Table D.71: Noisy optimization parameterized with SO(n) on $\mathrm{H}_3^{\oplus}\mathrm{SE}(2)$. Sensor and actuator noise set to $\sigma_c, \sigma_p = 0.10$. Hypotheses H_3 to H_4. True projection matrix and optimization results per iteration I_0 to I_4. Commutativity vector, unnormalized intrinsic conflict and commutativity delta.

Id	Hypothesis, $\Delta[\mathbb{O}_*]^{\ominus}$				
	H_0	H_1	H_2	H_3	H_4
I_0	0.0116	0.0299	0.0475	0.0507	0.0185
I_1	0.0413	0.0994	0.0287	0.0147	0.0291
I_2	0.0395	0.0053	0.0093	0.0127	0.0002
I_3	0.0537	0.0137	0.0537	0.2829	0.0000
I_4	0.0440	0.0902	0.0405	0.0491	0.0020
μ	0.0380	0.0477	0.0359	0.0820	0.0100
σ	0.0141	0.0394	0.0157	0.1017	0.0118
$\mu = 0.0427$, $\sigma = 0.0563$					

(a) Evaluation for $n_\theta = 10$.

Id	Hypothesis, $\Delta[\mathbb{O}_*]^{\ominus}$				
	H_0	H_1	H_2	H_3	H_4
I_0	0.0094	0.0219	0.2249	0.0390	0.0235
I_1	0.0391	0.0155	0.0311	0.0103	0.0252
I_2	0.0117	0.0059	0.0064	0.0321	0.0007
I_3	0.0650	0.0500	0.0240	0.3870	0.0000
I_4	0.0523	0.0998	0.0200	0.2524	0.0406
μ	0.0355	0.0386	0.0613	0.1442	0.0180
σ	0.0220	0.0339	0.0822	0.1498	0.0156
$\mu = 0.0595$, $\sigma = 0.0924$					

(b) Evaluation for $n_\theta = 20$.

Table D.72: Noisy optimization parameterized with SO(n) on $\mathrm{H}_3^{\ominus}\mathrm{SE}(2)$. Sensor and actuator noise set to $\sigma_c, \sigma_p = 0.10$. Commutativity delta $\Delta[\mathbb{O}_*]^{\ominus}$ per iteration I_0 to I_4. Mean and standard deviation of $\Delta[\mathbb{O}_*]^{\ominus}$ per hypothesis H_0 to H_4 and over all hypotheses.

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\textcircled{D}_*]^{\ominus}$
H_0	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 0.924 & -0.383 \\ 0.000 & 0.383 & 0.924 \end{bmatrix}$	$\textcircled{D}$ [0.38] -	
I_0	$\begin{bmatrix} -0.250 & 0.784 & 0.568 \\ -0.953 & -0.094 & -0.289 \\ -0.173 & -0.613 & 0.771 \end{bmatrix}$	$\textcircled{D}$ [0.64] -	$\Delta[\Xi_u]$ 0.5653 $\Delta[\textcircled{D}_*]^{\ominus}$ 0.2542
I_1	$\begin{bmatrix} -0.964 & -0.238 & 0.118 \\ -0.265 & 0.902 & -0.340 \\ -0.025 & -0.359 & -0.933 \end{bmatrix}$	$\textcircled{D}$ [0.36] -	$\Delta[\Xi_u]$ 0.5273 $\Delta[\textcircled{D}_*]^{\ominus}$ 0.0226
I_2	$\begin{bmatrix} 0.069 & 0.975 & 0.210 \\ 0.978 & -0.024 & -0.207 \\ -0.197 & 0.220 & -0.955 \end{bmatrix}$	$\textcircled{D}$ [0.30] -	$\Delta[\Xi_u]$ 0.6037 $\Delta[\textcircled{D}_*]^{\ominus}$ 0.0872
I_3	$\begin{bmatrix} -0.979 & -0.200 & -0.025 \\ 0.184 & -0.937 & 0.297 \\ -0.083 & 0.287 & 0.954 \end{bmatrix}$	$\textcircled{D}$ [0.30] -	$\Delta[\Xi_u]$ 0.5643 $\Delta[\textcircled{D}_*]^{\ominus}$ 0.0842
I_4	$\begin{bmatrix} 0.005 & 0.973 & -0.233 \\ 0.999 & -0.014 & -0.035 \\ -0.038 & -0.232 & -0.972 \end{bmatrix}$	$\textcircled{D}$ [0.24] -	$\Delta[\Xi_u]$ 0.5073 $\Delta[\textcircled{D}_*]^{\ominus}$ 0.1473

(a) Hypothesis H_0 for $n_\theta = 10$.

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\textcircled{D}_*]^{\ominus}$
H_0	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 0.924 & -0.383 \\ 0.000 & 0.383 & 0.924 \end{bmatrix}$	$\textcircled{D}$ [0.38] -	
I_0	$\begin{bmatrix} 0.771 & 0.593 & -0.231 \\ 0.630 & -0.763 & 0.144 \\ -0.091 & -0.256 & -0.962 \end{bmatrix}$	$\textcircled{D}$ [0.27] -	$\Delta[\Xi_u]$ 0.5356 $\Delta[\textcircled{D}_*]^{\ominus}$ 0.1108
I_1	$\begin{bmatrix} 0.428 & -0.853 & 0.300 \\ -0.903 & -0.388 & 0.186 \\ -0.042 & -0.350 & -0.936 \end{bmatrix}$	$\textcircled{D}$ [0.35] -	$\Delta[\Xi_u]$ 0.5171 $\Delta[\textcircled{D}_*]^{\ominus}$ 0.0301
I_2	$\begin{bmatrix} -0.994 & 0.000 & 0.111 \\ -0.032 & 0.959 & -0.283 \\ -0.106 & -0.285 & -0.953 \end{bmatrix}$	$\textcircled{D}$ [0.30] -	$\Delta[\Xi_u]$ 0.5797 $\Delta[\textcircled{D}_*]^{\ominus}$ 0.0784
I_3	$\begin{bmatrix} -0.991 & -0.104 & 0.083 \\ 0.128 & -0.923 & 0.363 \\ 0.039 & 0.371 & 0.928 \end{bmatrix}$	$\textcircled{D}$ [0.37] -	$\Delta[\Xi_u]$ 0.5607 $\Delta[\textcircled{D}_*]^{\ominus}$ 0.0098
I_4	$\begin{bmatrix} 0.532 & 0.809 & -0.250 \\ -0.832 & 0.554 & 0.024 \\ 0.158 & 0.195 & 0.968 \end{bmatrix}$	$\textcircled{D}$ [0.25] -	$\Delta[\Xi_u]$ 0.5135 $\Delta[\textcircled{D}_*]^{\ominus}$ 0.1315

(b) Hypothesis H_0 for $n_\theta = 20$.

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\textcircled{D}_*]^{\ominus}$
H_1	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 0.707 & -0.707 \\ 0.000 & 0.707 & 0.707 \end{bmatrix}$	$\textcircled{D}$ [0.71] -	
I_0	$\begin{bmatrix} 0.965 & 0.107 & 0.240 \\ 0.258 & -0.569 & -0.781 \\ 0.053 & 0.816 & -0.576 \end{bmatrix}$	$\textcircled{D}$ [0.82] -	$\Delta[\Xi_u]$ 0.5760 $\Delta[\textcircled{D}_*]^{\ominus}$ 0.1102
I_1	$\begin{bmatrix} -0.878 & 0.436 & 0.195 \\ 0.325 & 0.845 & -0.425 \\ -0.350 & -0.310 & -0.884 \end{bmatrix}$	$\textcircled{D}$ [0.47] -	$\Delta[\Xi_u]$ 0.5538 $\Delta[\textcircled{D}_*]^{\ominus}$ 0.2396
I_2	$\begin{bmatrix} -0.078 & 0.611 & 0.788 \\ 0.992 & 0.123 & 0.003 \\ -0.095 & 0.782 & -0.616 \end{bmatrix}$	$\textcircled{D}$ [0.79] -	$\Delta[\Xi_u]$ 0.5581 $\Delta[\textcircled{D}_*]^{\ominus}$ 0.0806
I_3	$\begin{bmatrix} -0.932 & -0.245 & -0.266 \\ -0.047 & 0.812 & -0.582 \\ 0.359 & -0.531 & -0.768 \end{bmatrix}$	$\textcircled{D}$ [0.64] -	$\Delta[\Xi_u]$ 0.5580 $\Delta[\textcircled{D}_*]^{\ominus}$ 0.0668
I_4	$\begin{bmatrix} 0.313 & 0.927 & -0.208 \\ 0.934 & -0.340 & -0.110 \\ -0.173 & -0.160 & -0.972 \end{bmatrix}$	$\textcircled{D}$ [0.24] -	$\Delta[\Xi_u]$ 0.5405 $\Delta[\textcircled{D}_*]^{\ominus}$ 0.4717

(c) Hypothesis H_1 for $n_\theta = 10$.

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\textcircled{D}_*]^{\ominus}$
H_1	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 0.707 & -0.707 \\ 0.000 & 0.707 & 0.707 \end{bmatrix}$	$\textcircled{D}$ [0.71] -	
I_0	$\begin{bmatrix} -0.964 & 0.009 & 0.267 \\ -0.139 & 0.838 & -0.528 \\ -0.229 & -0.546 & -0.806 \end{bmatrix}$	$\textcircled{D}$ [0.59] -	$\Delta[\Xi_u]$ 0.5584 $\Delta[\textcircled{D}_*]^{\ominus}$ 0.1154
I_1	$\begin{bmatrix} 0.731 & 0.552 & -0.402 \\ -0.682 & 0.624 & -0.382 \\ 0.040 & 0.553 & 0.832 \end{bmatrix}$	$\textcircled{D}$ [0.55] -	$\Delta[\Xi_u]$ 0.5495 $\Delta[\textcircled{D}_*]^{\ominus}$ 0.1530
I_2	$\begin{bmatrix} -0.995 & 0.064 & 0.072 \\ 0.003 & -0.725 & 0.689 \\ 0.096 & 0.686 & 0.721 \end{bmatrix}$	$\textcircled{D}$ [0.69] -	$\Delta[\Xi_u]$ 0.5497 $\Delta[\textcircled{D}_*]^{\ominus}$ 0.0146
I_3	$\begin{bmatrix} 0.288 & 0.609 & -0.739 \\ -0.957 & 0.151 & -0.248 \\ -0.039 & 0.779 & 0.626 \end{bmatrix}$	$\textcircled{D}$ [0.78] -	$\Delta[\Xi_u]$ 0.5350 $\Delta[\textcircled{D}_*]^{\ominus}$ 0.0728
I_4	$\begin{bmatrix} 0.499 & 0.753 & -0.429 \\ 0.769 & -0.613 & -0.180 \\ -0.399 & -0.240 & -0.885 \end{bmatrix}$	$\textcircled{D}$ [0.47] -	$\Delta[\Xi_u]$ 0.5351 $\Delta[\textcircled{D}_*]^{\ominus}$ 0.2415

(d) Hypothesis H_1 for $n_\theta = 20$.

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\textcircled{D}_*]^{\ominus}$
H_2	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 0.383 & -0.924 \\ 0.000 & 0.924 & 0.383 \end{bmatrix}$	$\textcircled{D}$ [0.92] -	
I_0	$\begin{bmatrix} -0.914 & 0.398 & 0.079 \\ -0.188 & -0.243 & -0.952 \\ -0.360 & -0.884 & 0.297 \end{bmatrix}$	$\textcircled{D}$ [0.95] -	$\Delta[\Xi_u]$ 0.5454 $\Delta[\textcircled{D}_*]^{\ominus}$ 0.0310
I_1	$\begin{bmatrix} -0.978 & 0.209 & -0.023 \\ 0.038 & 0.071 & -0.997 \\ -0.207 & -0.975 & -0.077 \end{bmatrix}$	$\textcircled{D}$ [1.00] -	$\Delta[\Xi_u]$ 0.5142 $\Delta[\textcircled{D}_*]^{\ominus}$ 0.0731
I_2	$\begin{bmatrix} 0.983 & -0.027 & 0.181 \\ 0.136 & -0.558 & -0.819 \\ 0.123 & 0.830 & -0.545 \end{bmatrix}$	$\textcircled{D}$ [0.84] -	$\Delta[\Xi_u]$ 0.5848 $\Delta[\textcircled{D}_*]^{\ominus}$ 0.0851
I_3	$\begin{bmatrix} 0.289 & -0.529 & 0.798 \\ -0.950 & -0.055 & 0.308 \\ -0.119 & -0.847 & -0.518 \end{bmatrix}$	$\textcircled{D}$ [0.86] -	$\Delta[\Xi_u]$ 0.5564 $\Delta[\textcircled{D}_*]^{\ominus}$ 0.0688
I_4	$\begin{bmatrix} 0.970 & 0.215 & -0.111 \\ -0.224 & 0.623 & -0.750 \\ -0.092 & 0.752 & 0.652 \end{bmatrix}$	$\textcircled{D}$ [0.76] -	$\Delta[\Xi_u]$ 0.5823 $\Delta[\textcircled{D}_*]^{\ominus}$ 0.1661

(e) Hypothesis H_2 for $n_\theta = 10$.

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\textcircled{D}_*]^{\ominus}$
H_2	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 0.383 & -0.924 \\ 0.000 & 0.924 & 0.383 \end{bmatrix}$	$\textcircled{D}$ [0.92] -	
I_0	$\begin{bmatrix} -0.233 & -0.423 & -0.876 \\ -0.971 & 0.049 & 0.234 \\ -0.056 & 0.905 & -0.422 \end{bmatrix}$	$\textcircled{D}$ [0.91] -	$\Delta[\Xi_u]$ 0.5366 $\Delta[\textcircled{D}_*]^{\ominus}$ 0.0175
I_1	$\begin{bmatrix} 0.905 & -0.195 & 0.378 \\ -0.372 & 0.070 & 0.926 \\ -0.207 & -0.978 & -0.009 \end{bmatrix}$	$\textcircled{D}$ [1.00] -	$\Delta[\Xi_u]$ 0.5273 $\Delta[\textcircled{D}_*]^{\ominus}$ 0.0761
I_2	$\begin{bmatrix} -0.527 & 0.441 & 0.727 \\ 0.844 & 0.166 & 0.510 \\ 0.104 & 0.882 & -0.460 \end{bmatrix}$	$\textcircled{D}$ [0.89] -	$\Delta[\Xi_u]$ 0.5821 $\Delta[\textcircled{D}_*]^{\ominus}$ 0.0358
I_3	$\begin{bmatrix} 0.306 & 0.473 & 0.826 \\ 0.943 & -0.266 & -0.198 \\ 0.126 & 0.840 & -0.527 \end{bmatrix}$	$\textcircled{D}$ [0.85] -	$\Delta[\Xi_u]$ 0.5605 $\Delta[\textcircled{D}_*]^{\ominus}$ 0.0742
I_4	$\begin{bmatrix} 0.249 & 0.472 & -0.845 \\ 0.952 & -0.281 & 0.124 \\ -0.179 & -0.835 & -0.520 \end{bmatrix}$	$\textcircled{D}$ [0.85] -	$\Delta[\Xi_u]$ 0.5793 $\Delta[\textcircled{D}_*]^{\ominus}$ 0.0695

(f) Hypothesis H_2 for $n_\theta = 20$.

Table D.73: Noisy optimization parameterized with SO(n) on $\mathrm{H}_3^{\oplus}\mathrm{SE}(2)$. Sensor and actuator noise set to $\sigma_c, \sigma_p = 0.25$. Hypotheses H_0 to H_2. True projection matrix and optimization results per iteration I_0 to I_4. Commutativity vector, unnormalized intrinsic conflict and commutativity delta.

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\oplus_*]^{\ominus}$
H_3	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 1.000 & 0.000 \\ 0.000 & 0.000 & 1.000 \end{bmatrix}$	ⓘ [0.00] -	
I_0	$\begin{bmatrix} 0.588 & 0.803 & 0.094 \\ 0.796 & -0.596 & 0.107 \\ 0.142 & 0.012 & -0.990 \end{bmatrix}$	ⓘ [0.14] -	$\Delta[\Xi_u]$ 0.5480 $\Delta[\oplus_*]^{\ominus}$ 0.1427
I_1	$\begin{bmatrix} -0.164 & -0.986 & 0.033 \\ -0.983 & 0.161 & -0.083 \\ 0.077 & -0.047 & -0.996 \end{bmatrix}$	ⓘ [0.09] -	$\Delta[\Xi_u]$ 0.5146 $\Delta[\oplus_*]^{\ominus}$ 0.0896
I_2	$\begin{bmatrix} -0.217 & 0.976 & 0.020 \\ 0.975 & 0.216 & 0.045 \\ 0.039 & 0.029 & -0.999 \end{bmatrix}$	ⓘ [0.05] -	$\Delta[\Xi_u]$ 0.4622 $\Delta[\oplus_*]^{\ominus}$ 0.0487
I_3	$\begin{bmatrix} -0.951 & -0.262 & -0.164 \\ 0.273 & -0.961 & -0.046 \\ -0.146 & -0.088 & 0.985 \end{bmatrix}$	ⓘ [0.17] -	$\Delta[\Xi_u]$ 0.5145 $\Delta[\oplus_*]^{\ominus}$ 0.1702
I_4	$\begin{bmatrix} 0.258 & 0.966 & -0.010 \\ 0.951 & -0.256 & -0.176 \\ -0.173 & 0.036 & -0.984 \end{bmatrix}$	ⓘ [0.18] -	$\Delta[\Xi_u]$ 0.4812 $\Delta[\oplus_*]^{\ominus}$ 0.1764

(a) Hypothesis H_3 for $n_\theta = 10$.

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\oplus_*]^{\ominus}$
H_3	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 1.000 & 0.000 \\ 0.000 & 0.000 & 1.000 \end{bmatrix}$	ⓘ [0.00] -	
I_0	$\begin{bmatrix} 0.392 & 0.912 & 0.123 \\ 0.919 & -0.383 & -0.096 \\ -0.040 & 0.151 & -0.988 \end{bmatrix}$	ⓘ [0.16] -	$\Delta[\Xi_u]$ 0.5463 $\Delta[\oplus_*]^{\ominus}$ 0.1561
I_1	$\begin{bmatrix} -0.472 & -0.880 & 0.050 \\ -0.880 & 0.468 & -0.079 \\ 0.046 & -0.081 & -0.996 \end{bmatrix}$	ⓘ [0.09] -	$\Delta[\Xi_u]$ 0.5122 $\Delta[\oplus_*]^{\ominus}$ 0.0931
I_2	$\begin{bmatrix} -0.463 & -0.886 & -0.042 \\ 0.886 & -0.464 & 0.021 \\ -0.038 & -0.027 & 0.999 \end{bmatrix}$	ⓘ [0.05] -	$\Delta[\Xi_u]$ 0.4643 $\Delta[\oplus_*]^{\ominus}$ 0.0469
I_3	$\begin{bmatrix} -0.842 & -0.538 & 0.025 \\ -0.535 & 0.830 & -0.156 \\ 0.063 & -0.145 & -0.987 \end{bmatrix}$	ⓘ [0.16] -	$\Delta[\Xi_u]$ 0.5167 $\Delta[\oplus_*]^{\ominus}$ 0.1582
I_4	$\begin{bmatrix} 0.788 & 0.613 & -0.066 \\ 0.609 & -0.790 & -0.067 \\ -0.093 & 0.013 & -0.996 \end{bmatrix}$	ⓘ [0.09] -	$\Delta[\Xi_u]$ 0.4826 $\Delta[\oplus_*]^{\ominus}$ 0.0942

(b) Hypothesis H_3 for $n_\theta = 20$.

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\oplus_*]^{\ominus}$
H_4	$\begin{bmatrix} 0.000 & 0.000 & 1.000 \\ 1.000 & 0.000 & 0.000 \\ 0.000 & 1.000 & 0.000 \end{bmatrix}$	ⓘ [1.00] -	
I_0	$\begin{bmatrix} 0.775 & 0.396 & 0.493 \\ -0.614 & 0.289 & 0.734 \\ 0.148 & -0.872 & 0.467 \end{bmatrix}$	ⓘ [0.88] -	$\Delta[\Xi_u]$ 0.6293 $\Delta[\oplus_*]^{\ominus}$ 0.1157
I_1	$\begin{bmatrix} 0.380 & 0.057 & -0.923 \\ -0.901 & -0.203 & -0.383 \\ -0.210 & 0.977 & -0.026 \end{bmatrix}$	ⓘ [1.00] -	$\Delta[\Xi_u]$ 0.6000 $\Delta[\oplus_*]^{\ominus}$ 0.0003
I_2	$\begin{bmatrix} -0.468 & 0.025 & 0.884 \\ 0.881 & 0.090 & 0.464 \\ -0.068 & 0.996 & -0.065 \end{bmatrix}$	ⓘ [1.00] -	$\Delta[\Xi_u]$ 0.5552 $\Delta[\oplus_*]^{\ominus}$ 0.0021
I_3	$\begin{bmatrix} -0.809 & -0.395 & -0.436 \\ -0.541 & 0.211 & 0.814 \\ -0.229 & 0.894 & -0.384 \end{bmatrix}$	ⓘ [0.92] -	$\Delta[\Xi_u]$ 0.5652 $\Delta[\oplus_*]^{\ominus}$ 0.0767
I_4	$\begin{bmatrix} -0.707 & 0.615 & 0.350 \\ 0.556 & 0.177 & 0.812 \\ 0.438 & 0.769 & -0.466 \end{bmatrix}$	ⓘ [0.88] -	$\Delta[\Xi_u]$ 0.6035 $\Delta[\oplus_*]^{\ominus}$ 0.1154

(c) Hypothesis H_4 for $n_\theta = 10$.

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\oplus_*]^{\ominus}$
H_4	$\begin{bmatrix} 0.000 & 0.000 & 1.000 \\ 1.000 & 0.000 & 0.000 \\ 0.000 & 1.000 & 0.000 \end{bmatrix}$	ⓘ [1.00] -	
I_0	$\begin{bmatrix} -0.351 & -0.602 & -0.717 \\ -0.910 & 0.038 & 0.413 \\ -0.222 & 0.797 & -0.561 \end{bmatrix}$	ⓘ [0.83] -	$\Delta[\Xi_u]$ 0.6192 $\Delta[\oplus_*]^{\ominus}$ 0.1725
I_1	$\begin{bmatrix} 0.961 & 0.056 & -0.272 \\ 0.277 & -0.119 & 0.953 \\ 0.021 & -0.991 & -0.130 \end{bmatrix}$	ⓘ [0.99] -	$\Delta[\Xi_u]$ 0.6044 $\Delta[\oplus_*]^{\ominus}$ 0.0085
I_2	$\begin{bmatrix} -0.031 & 0.050 & 0.998 \\ 0.997 & 0.070 & 0.027 \\ -0.069 & 0.996 & -0.052 \end{bmatrix}$	ⓘ [1.00] -	$\Delta[\Xi_u]$ 0.5586 $\Delta[\oplus_*]^{\ominus}$ 0.0014
I_3	$\begin{bmatrix} -0.264 & -0.121 & 0.957 \\ -0.946 & -0.163 & -0.281 \\ 0.190 & -0.979 & -0.072 \end{bmatrix}$	ⓘ [1.00] -	$\Delta[\Xi_u]$ 0.5655 $\Delta[\oplus_*]^{\ominus}$ 0.0026
I_4	$\begin{bmatrix} 0.762 & 0.124 & 0.635 \\ -0.631 & 0.358 & 0.688 \\ -0.142 & -0.925 & 0.351 \end{bmatrix}$	ⓘ [0.94] -	$\Delta[\Xi_u]$ 0.5981 $\Delta[\oplus_*]^{\ominus}$ 0.0637

(d) Hypothesis H_4 for $n_\theta = 20$.

Table D.74: Noisy optimization parameterized with SO(n) on $\mathrm{H}_3^{\oplus}\mathrm{SE}(2)$. Sensor and actuator noise set to $\sigma_c, \sigma_p = 0.25$. Hypotheses H_3 to H_4. True projection matrix and optimization results per iteration I_0 to I_4. Commutativity vector, unnormalized intrinsic conflict and commutativity delta.

Id	Hypothesis, $\Delta[\oplus_*]^{\ominus}$				
	H_0	H_1	H_2	H_3	H_4
I_0	0.2542	0.1102	0.0310	0.1427	0.1157
I_1	0.0226	0.2396	0.0731	0.0896	0.0003
I_2	0.0872	0.0806	0.0851	0.0487	0.0021
I_3	0.0842	0.0668	0.0688	0.1702	0.0767
I_4	0.1473	0.4717	0.1661	0.1764	0.1154
μ	0.1191	0.1938	0.0848	0.1255	0.0620
σ	0.0782	0.1518	0.0445	0.0491	0.0517
$\mu = 0.1171$, $\sigma = 0.0982$					

(a) Evaluation for $n_\theta = 10$.

Id	Hypothesis, $\Delta[\oplus_*]^{\ominus}$				
	H_0	H_1	H_2	H_3	H_4
I_0	0.1108	0.1154	0.0175	0.1561	0.1725
I_1	0.0301	0.1530	0.0761	0.0931	0.0085
I_2	0.0784	0.0146	0.0358	0.0469	0.0014
I_3	0.0098	0.0728	0.0742	0.1582	0.0026
I_4	0.1315	0.2415	0.0695	0.0942	0.0637
μ	0.0721	0.1195	0.0546	0.1097	0.0497
σ	0.0463	0.0764	0.0237	0.0423	0.0656
$\mu = 0.0811$, $\sigma = 0.0624$					

(b) Evaluation for $n_\theta = 20$.

Table D.75: Noisy optimization parameterized with SO(n) on $\mathrm{H}_3^{\ominus}\mathrm{SE}(2)$. Sensor and actuator noise set to $\sigma_c, \sigma_p = 0.25$. Commutativity delta $\Delta[\oplus_*]^{\ominus}$ per iteration I_0 to I_4. Mean and standard deviation of $\Delta[\oplus_*]^{\ominus}$ per hypothesis H_0 to H_4 and over all hypotheses.

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\textcircled{D}_*]^{\ominus}$
H_0	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 0.924 & -0.383 \\ 0.000 & 0.383 & 0.924 \end{bmatrix}$	$\textcircled{D}$ [0.38] -	
I_0	$\begin{bmatrix} 0.473 & 0.879 & 0.067 \\ -0.844 & 0.473 & -0.254 \\ -0.255 & 0.064 & 0.965 \end{bmatrix}$	$\textcircled{D}$ [0.26] -	$\Delta[\Xi_u]$ 1.6229 $\Delta[\textcircled{D}_*]^{\ominus}$ 0.1198
I_1	$\begin{bmatrix} -0.030 & -0.991 & 0.133 \\ -0.972 & -0.003 & -0.237 \\ 0.235 & -0.136 & -0.962 \end{bmatrix}$	$\textcircled{D}$ [0.27] -	$\Delta[\Xi_u]$ 1.5689 $\Delta[\textcircled{D}_*]^{\ominus}$ 0.1111
I_2	$\begin{bmatrix} -0.257 & 0.932 & -0.257 \\ 0.959 & 0.279 & 0.055 \\ 0.123 & -0.232 & -0.965 \end{bmatrix}$	$\textcircled{D}$ [0.26] -	$\Delta[\Xi_u]$ 1.5724 $\Delta[\textcircled{D}_*]^{\ominus}$ 0.1199
I_3	$\begin{bmatrix} -0.978 & -0.134 & -0.159 \\ 0.154 & -0.981 & -0.117 \\ -0.140 & -0.139 & 0.980 \end{bmatrix}$	$\textcircled{D}$ [0.20] -	$\Delta[\Xi_u]$ 1.5785 $\Delta[\textcircled{D}_*]^{\ominus}$ 0.1852
I_4	$\begin{bmatrix} -0.709 & 0.695 & -0.122 \\ -0.700 & -0.714 & 0.002 \\ -0.086 & 0.087 & 0.993 \end{bmatrix}$	$\textcircled{D}$ [0.12] -	$\Delta[\Xi_u]$ 1.5240 $\Delta[\textcircled{D}_*]^{\ominus}$ 0.2608

(a) Hypothesis H_0 for $n_\theta = 10$.

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\textcircled{D}_*]^{\ominus}$
H_0	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 0.924 & -0.383 \\ 0.000 & 0.383 & 0.924 \end{bmatrix}$	$\textcircled{D}$ [0.38] -	
I_0	$\begin{bmatrix} 0.407 & 0.890 & -0.207 \\ 0.897 & -0.346 & 0.277 \\ 0.175 & -0.298 & -0.938 \end{bmatrix}$	$\textcircled{D}$ [0.35] -	$\Delta[\Xi_u]$ 1.6205 $\Delta[\textcircled{D}_*]^{\ominus}$ 0.0370
I_1	$\begin{bmatrix} -0.831 & 0.556 & -0.018 \\ 0.545 & 0.821 & 0.170 \\ 0.109 & 0.131 & -0.985 \end{bmatrix}$	$\textcircled{D}$ [0.17] -	$\Delta[\Xi_u]$ 1.5804 $\Delta[\textcircled{D}_*]^{\ominus}$ 0.2121
I_2	$\begin{bmatrix} -0.534 & 0.830 & 0.163 \\ 0.812 & 0.557 & -0.174 \\ -0.235 & 0.040 & -0.971 \end{bmatrix}$	$\textcircled{D}$ [0.24] -	$\Delta[\Xi_u]$ 1.5568 $\Delta[\textcircled{D}_*]^{\ominus}$ 0.1442
I_3	$\begin{bmatrix} -0.893 & -0.444 & -0.077 \\ 0.439 & -0.895 & 0.076 \\ -0.103 & 0.034 & 0.994 \end{bmatrix}$	$\textcircled{D}$ [0.11] -	$\Delta[\Xi_u]$ 1.5824 $\Delta[\textcircled{D}_*]^{\ominus}$ 0.2743
I_4	$\begin{bmatrix} 0.800 & 0.533 & -0.276 \\ 0.538 & -0.841 & -0.064 \\ -0.266 & -0.098 & -0.959 \end{bmatrix}$	$\textcircled{D}$ [0.28] -	$\Delta[\Xi_u]$ 1.5575 $\Delta[\textcircled{D}_*]^{\ominus}$ 0.0990

(b) Hypothesis H_0 for $n_\theta = 20$.

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\textcircled{D}_*]^{\ominus}$
H_1	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 0.707 & -0.707 \\ 0.000 & 0.707 & 0.707 \end{bmatrix}$	$\textcircled{D}$ [0.71] -	
I_0	$\begin{bmatrix} 0.974 & 0.186 & -0.127 \\ -0.137 & 0.936 & 0.324 \\ 0.179 & -0.299 & 0.937 \end{bmatrix}$	$\textcircled{D}$ [0.35] -	$\Delta[\Xi_u]$ 1.6241 $\Delta[\textcircled{D}_*]^{\ominus}$ 0.3590
I_1	$\begin{bmatrix} 0.589 & -0.683 & -0.431 \\ -0.787 & -0.605 & -0.117 \\ -0.181 & 0.409 & -0.895 \end{bmatrix}$	$\textcircled{D}$ [0.45] -	$\Delta[\Xi_u]$ 1.5636 $\Delta[\textcircled{D}_*]^{\ominus}$ 0.2604
I_2	$\begin{bmatrix} 0.142 & 0.273 & -0.951 \\ -0.939 & -0.265 & -0.217 \\ -0.312 & 0.925 & 0.219 \end{bmatrix}$	$\textcircled{D}$ [0.98] -	$\Delta[\Xi_u]$ 1.6688 $\Delta[\textcircled{D}_*]^{\ominus}$ 0.2686
I_3	$\begin{bmatrix} 0.353 & -0.777 & -0.520 \\ 0.915 & 0.403 & 0.019 \\ 0.195 & -0.483 & 0.854 \end{bmatrix}$	$\textcircled{D}$ [0.52] -	$\Delta[\Xi_u]$ 1.6440 $\Delta[\textcircled{D}_*]^{\ominus}$ 0.1863
I_4	$\begin{bmatrix} -0.905 & 0.040 & 0.423 \\ 0.011 & 0.998 & -0.069 \\ -0.425 & -0.058 & -0.903 \end{bmatrix}$	$\textcircled{D}$ [0.43] -	$\Delta[\Xi_u]$ 1.5817 $\Delta[\textcircled{D}_*]^{\ominus}$ 0.2785

(c) Hypothesis H_1 for $n_\theta = 10$.

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\textcircled{D}_*]^{\ominus}$
H_1	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 0.707 & -0.707 \\ 0.000 & 0.707 & 0.707 \end{bmatrix}$	$\textcircled{D}$ [0.71] -	
I_0	$\begin{bmatrix} 0.602 & -0.714 & -0.358 \\ 0.774 & 0.632 & 0.041 \\ 0.197 & -0.302 & 0.933 \end{bmatrix}$	$\textcircled{D}$ [0.36] -	$\Delta[\Xi_u]$ 1.6271 $\Delta[\textcircled{D}_*]^{\ominus}$ 0.3463
I_1	$\begin{bmatrix} 0.647 & -0.559 & 0.519 \\ -0.691 & -0.718 & 0.088 \\ 0.323 & -0.415 & -0.850 \end{bmatrix}$	$\textcircled{D}$ [0.53] -	$\Delta[\Xi_u]$ 1.5535 $\Delta[\textcircled{D}_*]^{\ominus}$ 0.1809
I_2	$\begin{bmatrix} -0.038 & 0.963 & 0.266 \\ 0.999 & 0.038 & 0.007 \\ -0.003 & 0.266 & -0.964 \end{bmatrix}$	$\textcircled{D}$ [0.27] -	$\Delta[\Xi_u]$ 1.6493 $\Delta[\textcircled{D}_*]^{\ominus}$ 0.4411
I_3	$\begin{bmatrix} -0.790 & 0.469 & -0.394 \\ 0.578 & 0.784 & -0.225 \\ 0.203 & -0.406 & -0.891 \end{bmatrix}$	$\textcircled{D}$ [0.45] -	$\Delta[\Xi_u]$ 1.6001 $\Delta[\textcircled{D}_*]^{\ominus}$ 0.2536
I_4	$\begin{bmatrix} 0.989 & 0.129 & -0.074 \\ 0.117 & -0.982 & -0.147 \\ -0.091 & 0.137 & -0.986 \end{bmatrix}$	$\textcircled{D}$ [0.16] -	$\Delta[\Xi_u]$ 1.5896 $\Delta[\textcircled{D}_*]^{\ominus}$ 0.5428

(d) Hypothesis H_1 for $n_\theta = 20$.

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\textcircled{D}_*]^{\ominus}$
H_2	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 0.383 & -0.924 \\ 0.000 & 0.924 & 0.383 \end{bmatrix}$	$\textcircled{D}$ [0.92] -	
I_0	$\begin{bmatrix} 0.541 & 0.827 & -0.156 \\ 0.824 & -0.483 & 0.296 \\ 0.170 & -0.289 & -0.942 \end{bmatrix}$	$\textcircled{D}$ [0.33] -	$\Delta[\Xi_u]$ 1.6963 $\Delta[\textcircled{D}_*]^{\ominus}$ 0.5890
I_1	$\begin{bmatrix} -0.109 & -0.993 & 0.038 \\ -0.966 & 0.097 & -0.242 \\ 0.237 & -0.063 & -0.970 \end{bmatrix}$	$\textcircled{D}$ [0.24] -	$\Delta[\Xi_u]$ 1.5181 $\Delta[\textcircled{D}_*]^{\ominus}$ 0.6792
I_2	$\begin{bmatrix} -0.240 & 0.965 & 0.106 \\ 0.969 & 0.231 & 0.087 \\ 0.060 & 0.123 & -0.991 \end{bmatrix}$	$\textcircled{D}$ [0.14] -	$\Delta[\Xi_u]$ 1.6235 $\Delta[\textcircled{D}_*]^{\ominus}$ 0.7869
I_3	$\begin{bmatrix} 0.147 & -0.987 & -0.061 \\ 0.984 & 0.152 & -0.092 \\ 0.100 & -0.046 & 0.994 \end{bmatrix}$	$\textcircled{D}$ [0.11] -	$\Delta[\Xi_u]$ 1.5684 $\Delta[\textcircled{D}_*]^{\ominus}$ 0.8134
I_4	$\begin{bmatrix} 0.022 & 0.888 & 0.460 \\ 0.993 & -0.071 & 0.090 \\ 0.113 & 0.455 & -0.883 \end{bmatrix}$	$\textcircled{D}$ [0.47] -	$\Delta[\Xi_u]$ 1.5955 $\Delta[\textcircled{D}_*]^{\ominus}$ 0.4548

(e) Hypothesis H_2 for $n_\theta = 10$.

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\textcircled{D}_*]^{\ominus}$
H_2	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 0.383 & -0.924 \\ 0.000 & 0.924 & 0.383 \end{bmatrix}$	$\textcircled{D}$ [0.92] -	
I_0	$\begin{bmatrix} 0.702 & 0.712 & -0.012 \\ 0.705 & -0.698 & -0.125 \\ -0.097 & 0.079 & -0.992 \end{bmatrix}$	$\textcircled{D}$ [0.13] -	$\Delta[\Xi_u]$ 1.6748 $\Delta[\textcircled{D}_*]^{\ominus}$ 0.7984
I_1	$\begin{bmatrix} -0.940 & 0.322 & -0.114 \\ 0.328 & 0.944 & -0.030 \\ 0.098 & -0.066 & -0.993 \end{bmatrix}$	$\textcircled{D}$ [0.12] -	$\Delta[\Xi_u]$ 1.5272 $\Delta[\textcircled{D}_*]^{\ominus}$ 0.8057
I_2	$\begin{bmatrix} -0.047 & 0.981 & 0.188 \\ 0.979 & 0.009 & 0.202 \\ 0.197 & 0.194 & -0.961 \end{bmatrix}$	$\textcircled{D}$ [0.28] -	$\Delta[\Xi_u]$ 1.6346 $\Delta[\textcircled{D}_*]^{\ominus}$ 0.6479
I_3	$\begin{bmatrix} -0.824 & -0.567 & -0.008 \\ 0.563 & -0.817 & -0.124 \\ 0.064 & -0.107 & 0.992 \end{bmatrix}$	$\textcircled{D}$ [0.12] -	$\Delta[\Xi_u]$ 1.5559 $\Delta[\textcircled{D}_*]^{\ominus}$ 0.7995
I_4	$\begin{bmatrix} 0.995 & -0.097 & -0.035 \\ 0.074 & 0.904 & -0.420 \\ 0.073 & 0.415 & 0.907 \end{bmatrix}$	$\textcircled{D}$ [0.42] -	$\Delta[\Xi_u]$ 1.5798 $\Delta[\textcircled{D}_*]^{\ominus}$ 0.5022

(f) Hypothesis H_2 for $n_\theta = 20$.

Table D.76: Noisy optimization parameterized with SO(n) on $\mathrm{H}_3^{\oplus}\mathrm{SE}(2)$. Sensor and actuator noise set to $\sigma_c, \sigma_p = 0.50$. Hypotheses H_0 to H_2. True projection matrix and optimization results per iteration I_0 to I_4. Commutativity vector, unnormalized intrinsic conflict and commutativity delta.

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\textcircled{D}_*]^{\ominus}$
H_3	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 1.000 & 0.000 \\ 0.000 & 0.000 & 1.000 \end{bmatrix}$	$\textcircled{D}$ [0.00] -	
I_0	$\begin{bmatrix} 0.618 & 0.780 & -0.093 \\ 0.774 & -0.625 & -0.099 \\ -0.135 & -0.011 & -0.991 \end{bmatrix}$	$\textcircled{D}$ [0.14] -	$\Delta[\Xi_u]$ 1.5763 $\Delta[\textcircled{D}_*]^{\ominus}$ 0.1357
I_1	$\begin{bmatrix} -0.049 & -0.998 & 0.043 \\ -0.999 & 0.050 & 0.019 \\ -0.021 & -0.042 & -0.999 \end{bmatrix}$	$\textcircled{D}$ [0.05] -	$\Delta[\Xi_u]$ 1.4323 $\Delta[\textcircled{D}_*]^{\ominus}$ 0.0471
I_2	$\begin{bmatrix} -0.005 & 0.999 & 0.041 \\ 1.000 & 0.004 & 0.007 \\ 0.007 & 0.041 & -0.999 \end{bmatrix}$	$\textcircled{D}$ [0.04] -	$\Delta[\Xi_u]$ 1.4924 $\Delta[\textcircled{D}_*]^{\ominus}$ 0.0417
I_3	$\begin{bmatrix} -0.975 & -0.201 & 0.094 \\ 0.204 & -0.979 & 0.023 \\ 0.088 & 0.042 & 0.995 \end{bmatrix}$	$\textcircled{D}$ [0.10] -	$\Delta[\Xi_u]$ 1.5521 $\Delta[\textcircled{D}_*]^{\ominus}$ 0.0970
I_4	$\begin{bmatrix} 0.475 & -0.879 & 0.049 \\ 0.880 & 0.476 & 0.002 \\ -0.025 & 0.042 & 0.999 \end{bmatrix}$	$\textcircled{D}$ [0.05] -	$\Delta[\Xi_u]$ 1.4403 $\Delta[\textcircled{D}_*]^{\ominus}$ 0.0491

(a) Hypothesis H_3 for $n_\theta = 10$.

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\textcircled{D}_*]^{\ominus}$
H_3	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 1.000 & 0.000 \\ 0.000 & 0.000 & 1.000 \end{bmatrix}$	$\textcircled{D}$ [0.00] -	
I_0	$\begin{bmatrix} 0.606 & 0.793 & -0.060 \\ 0.790 & -0.609 & -0.069 \\ -0.092 & -0.005 & -0.996 \end{bmatrix}$	$\textcircled{D}$ [0.09] -	$\Delta[\Xi_u]$ 1.5683 $\Delta[\textcircled{D}_*]^{\ominus}$ 0.0917
I_1	$\begin{bmatrix} -0.161 & -0.984 & 0.075 \\ -0.987 & 0.160 & -0.025 \\ 0.012 & -0.078 & -0.997 \end{bmatrix}$	$\textcircled{D}$ [0.08] -	$\Delta[\Xi_u]$ 1.4452 $\Delta[\textcircled{D}_*]^{\ominus}$ 0.0792
I_2	$\begin{bmatrix} -0.371 & -0.929 & 0.018 \\ 0.922 & -0.366 & 0.126 \\ -0.111 & 0.063 & 0.992 \end{bmatrix}$	$\textcircled{D}$ [0.13] -	$\Delta[\Xi_u]$ 1.4914 $\Delta[\textcircled{D}_*]^{\ominus}$ 0.1276
I_3	$\begin{bmatrix} -0.977 & -0.197 & 0.086 \\ 0.201 & -0.979 & 0.040 \\ 0.076 & 0.057 & 0.995 \end{bmatrix}$	$\textcircled{D}$ [0.09] -	$\Delta[\Xi_u]$ 1.5517 $\Delta[\textcircled{D}_*]^{\ominus}$ 0.0949
I_4	$\begin{bmatrix} 0.879 & 0.474 & -0.058 \\ 0.470 & -0.880 & -0.064 \\ -0.082 & 0.029 & -0.996 \end{bmatrix}$	$\textcircled{D}$ [0.09] -	$\Delta[\Xi_u]$ 1.4346 $\Delta[\textcircled{D}_*]^{\ominus}$ 0.0866

(b) Hypothesis H_3 for $n_\theta = 20$.

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\textcircled{D}_*]^{\ominus}$
H_4	$\begin{bmatrix} 0.000 & 0.000 & 1.000 \\ 1.000 & 0.000 & 0.000 \\ 0.000 & 1.000 & 0.000 \end{bmatrix}$	$\textcircled{D}$ [1.00] -	
I_0	$\begin{bmatrix} 0.690 & 0.722 & 0.063 \\ 0.566 & -0.483 & -0.668 \\ -0.451 & 0.496 & -0.742 \end{bmatrix}$	$\textcircled{D}$ [0.67] -	$\Delta[\Xi_u]$ 1.7113 $\Delta[\textcircled{D}_*]^{\ominus}$ 0.3296
I_1	$\begin{bmatrix} 0.914 & 0.327 & 0.240 \\ 0.183 & 0.196 & -0.963 \\ -0.362 & 0.924 & 0.120 \end{bmatrix}$	$\textcircled{D}$ [0.99] -	$\Delta[\Xi_u]$ 1.6826 $\Delta[\textcircled{D}_*]^{\ominus}$ 0.0072
I_2	$\begin{bmatrix} 0.674 & 0.150 & 0.724 \\ 0.716 & -0.373 & -0.590 \\ 0.181 & 0.916 & -0.359 \end{bmatrix}$	$\textcircled{D}$ [0.93] -	$\Delta[\Xi_u]$ 1.7043 $\Delta[\textcircled{D}_*]^{\ominus}$ 0.0667
I_3	$\begin{bmatrix} -0.010 & -0.096 & -0.995 \\ 0.992 & 0.125 & -0.022 \\ 0.127 & -0.987 & 0.094 \end{bmatrix}$	$\textcircled{D}$ [1.00] -	$\Delta[\Xi_u]$ 1.6765 $\Delta[\textcircled{D}_*]^{\ominus}$ 0.0044
I_4	$\begin{bmatrix} -0.174 & 0.697 & -0.695 \\ 0.981 & 0.058 & -0.187 \\ -0.090 & -0.714 & -0.694 \end{bmatrix}$	$\textcircled{D}$ [0.72] -	$\Delta[\Xi_u]$ 1.6008 $\Delta[\textcircled{D}_*]^{\ominus}$ 0.2802

(c) Hypothesis H_4 for $n_\theta = 10$.

Id	Projection Matrix	Commutativity	$\Delta[\Xi_u]$, $\Delta[\textcircled{D}_*]^{\ominus}$
H_4	$\begin{bmatrix} 0.000 & 0.000 & 1.000 \\ 1.000 & 0.000 & 0.000 \\ 0.000 & 1.000 & 0.000 \end{bmatrix}$	$\textcircled{D}$ [1.00] -	
I_0	$\begin{bmatrix} 0.884 & 0.275 & -0.378 \\ -0.037 & 0.847 & 0.530 \\ 0.466 & -0.454 & 0.759 \end{bmatrix}$	$\textcircled{D}$ [0.65] -	$\Delta[\Xi_u]$ 1.7025 $\Delta[\textcircled{D}_*]^{\ominus}$ 0.3493
I_1	$\begin{bmatrix} -0.007 & -0.999 & -0.037 \\ -0.997 & 0.010 & -0.074 \\ 0.075 & 0.036 & -0.997 \end{bmatrix}$	$\textcircled{D}$ [0.08] -	$\Delta[\Xi_u]$ 1.6884 $\Delta[\textcircled{D}_*]^{\ominus}$ 0.9170
I_2	$\begin{bmatrix} -0.405 & 0.104 & 0.908 \\ 0.896 & -0.150 & 0.417 \\ 0.180 & 0.983 & -0.032 \end{bmatrix}$	$\textcircled{D}$ [1.00] -	$\Delta[\Xi_u]$ 1.6798 $\Delta[\textcircled{D}_*]^{\ominus}$ 0.0005
I_3	$\begin{bmatrix} -0.913 & -0.199 & -0.355 \\ -0.328 & 0.876 & 0.354 \\ 0.241 & 0.440 & -0.865 \end{bmatrix}$	$\textcircled{D}$ [0.50] -	$\Delta[\Xi_u]$ 1.6574 $\Delta[\textcircled{D}_*]^{\ominus}$ 0.4984
I_4	$\begin{bmatrix} -0.551 & 0.418 & -0.723 \\ 0.833 & 0.332 & -0.443 \\ 0.055 & -0.846 & -0.531 \end{bmatrix}$	$\textcircled{D}$ [0.85] -	$\Delta[\Xi_u]$ 1.5855 $\Delta[\textcircled{D}_*]^{\ominus}$ 0.1524

(d) Hypothesis H_4 for $n_\theta = 20$.

Table D.77: Noisy optimization parameterized with SO(n) on $\mathrm{H}_3^{\oplus}\mathrm{SE}(2)$. Sensor and actuator noise set to $\sigma_c, \sigma_p = 0.50$. Hypotheses H_3 to H_4. True projection matrix and optimization results per iteration I_0 to I_4. Commutativity vector, unnormalized intrinsic conflict and commutativity delta.

Id	Hypothesis, $\Delta[\textcircled{D}_*]^{\ominus}$				
	H_0	H_1	H_2	H_3	H_4
I_0	0.1198	0.3590	0.5890	0.1357	0.3296
I_1	0.1111	0.2604	0.6792	0.0471	0.0072
I_2	0.1199	0.2686	0.7869	0.0417	0.0667
I_3	0.1852	0.1863	0.8134	0.0970	0.0044
I_4	0.2608	0.2785	0.4548	0.0491	0.2802
μ	0.1594	0.2706	0.6647	0.0741	0.1376
σ	0.0573	0.0549	0.1319	0.0367	0.1393
$\mu = 0.2613$, $\sigma = 0.2363$					

(a) Evaluation for $n_\theta = 10$.

Id	Hypothesis, $\Delta[\textcircled{D}_*]^{\ominus}$				
	H_0	H_1	H_2	H_3	H_4
I_0	0.0370	0.3463	0.7984	0.0917	0.3493
I_1	0.2121	0.1809	0.8057	0.0792	0.9170
I_2	0.1442	0.4411	0.6479	0.1276	0.0005
I_3	0.2743	0.2536	0.7995	0.0949	0.4984
I_4	0.0990	0.5428	0.5022	0.0866	0.1524
μ	0.1533	0.3529	0.7107	0.0960	0.3835
σ	0.0833	0.1291	0.1200	0.0167	0.3159
$\mu = 0.3393$, $\sigma = 0.2784$					

(b) Evaluation for $n_\theta = 20$.

Table D.78: Noisy optimization parameterized with SO(n) on $\mathrm{H}_3^{\ominus}\mathrm{SE}(2)$. Sensor and actuator noise set to $\sigma_c, \sigma_p = 0.50$. Commutativity delta $\Delta[\textcircled{D}_*]^{\ominus}$ per iteration I_0 to I_4. Mean and standard deviation of $\Delta[\textcircled{D}_*]^{\ominus}$ per hypothesis H_0 to H_4 and over all hypotheses.

Bibliography

[Amato and Song, 2002] Amato, N. and Song, G. (2002). Using motion planning to study protein folding pathways. *Journal of computational biology : A journal of computational molecular cell biology*, 9(2):149–68.

[Aoki et al., 2016] Aoki, T., Nakamura, T., and Nagai, T. (2016). Learning of motor control from motor babbling. *IFAC-PapersOnLine*, 49(19):154 – 158. 13th IFAC Symposium on Analysis, Design, and Evaluation ofHuman-Machine Systems HMS 2016.

[Arieşanu, 2015] Arieşanu, C. (2015). A drift-free left invariant control system on the lie group so(3) x r-3 x r-3. *Mathematical Problems in Engineering*, 2015:1–9.

[Arleo and Gerstner, 2001] Arleo, A. and Gerstner, W. (2001). Hippocampal spatial model for state space representation in robotic reinforcement learning. In Wiering, M. A., editor, *Proceedings of the Fifth European Workshop on Reinforcement Learning*, pages 1–3. CKI, Utrecht University.

[Baillargeon, 1994] Baillargeon, R. (1994). How do infants learn about the physical world? *Current Directions in Psychological Science*, 3(5):133–140.

[Baillargeon et al., 2010] Baillargeon, R., Li, J., Gertner, Y., and Wu, D. (2010). *The Wiley-Blackwell Handbook of Child Cognitive Development*, chapter How Do Infants Reason about Physical Events?, pages 11 – 48. Wiley-Blackwell.

[Bajracharya et al., 2008] Bajracharya, M., Maimone, M. W., and Helmick, D. (2008). Autonomy for mars rovers: Past, present, and future. *Computer*, 41(12):44–50.

[Balke and Gilbert, 2014] Balke, T. and Gilbert, N. (2014). How do agents make decisions? a survey. *JASSS*, 17(4):13.

[Banquet et al., 2005] Banquet, J. P., Gaussier, P., Quoy, M., Revel, A., and Burnod, Y. (2005). A hierarchy of associations in hippocampo-cortical systems: Cognitive maps and navigation strategies. *Neural Computation*, 17(6):1339–1384.

[Berenson et al., 2009] Berenson, D., Srinivasa, S. S., Ferguson, D., and Kuffner, J. J. (2009). Manipulation planning on constraint manifolds. In *2009 IEEE International Conference on Robotics and Automation*, pages 625–632.

[Bichucher et al., 2015] Bichucher, V., Walls, J. M., Ozog, P., Skinner, K. A., and Eustice, R. M. (2015). Bathymetric factor graph slam with sparse point cloud alignment. In *OCEANS 2015 - MTS/IEEE Washington*, pages 1–7.

[Bishop, 2006] Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer-Verlag New York.

[Bjerknes et al., 2014] Bjerknes, T., Moser, E., and Moser, M.-B. (2014). Representation of geometric borders in the developing rat. *Neuron*, 82(1):71 – 78.

[Bloch et al., 2007] Bloch, A., Baillieul, J., Crouch, P., and Marsden, J. (2007). *Nonholonomic Mechanics and Control*. Interdisciplinary Applied Mathematics. Springer New York.

[Bongard et al., 2006] Bongard, J., Zykov, V., and Lipson, H. (2006). Resilient machines through continuous self-modeling. *Science*, 314(5802):1118–1121.

[Brodal, 2004] Brodal, P. (2004). *The Central Nervous System*. Oxford University Press, 3rd edition.

[Broomhead and Lowe, 1988] Broomhead, D. and Lowe, D. (1988). Radial basis functions, multi-variable functional interpolation and adaptive networks. *ROYAL SIGNALS AND RADAR ESTABLISHMENT MALVERN (UNITED KINGDOM)*, RSRE-MEMO-4148.

[Buhrmann et al., 2013] Buhrmann, T., Di Paolo, E., and Barandiaran, X. (2013). A dynamical systems account of sensorimotor contingencies. *Frontiers in psychology*, 4:285.

[Caligiore et al., 2008] Caligiore, D., Ferrauto, T., Parisi, D., Accornero, N., Capozza, M., and Baldassarre, G. (2008). Using motor babbling and hebb rules for modeling the development of reaching with obstacles and grasping. In *International Conference on Cognitive Systems*.

[Censi and Murray, 2015] Censi, A. and Murray, R. (2015). Bootstrapping bilinear models of simple vehicles. *The International Journal of Robotics Research*, 34(8):1087–1113.

[Chahl et al., 2003] Chahl, J., Thakoor, S., Bouffant, N., Stange, G., Srinivasan, M., Hine, B., and Zornetzer, S. (2003). Bioinspired engineering of exploration systems: A horizon sensor/attitude reference system based on the dragonfly ocelli for mars exploration applications. *Journal of Robotic Systems*, 20(1):35 – 42.

[Choset et al., 2005] Choset, H., Lynch, K., Hutchinson, S., Kantor, G., Burgard, W., Kavraki, L., and Thrun, S. (2005). *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press.

[Choset and Nagatani, 2001] Choset, H. and Nagatani, K. (2001). Topological simultaneous localization and mapping (slam): toward exact localization without explicit localization. *IEEE Transactions on Robotics and Automation*, 17(2):125–137.

[Chutia et al., 2017] Chutia, S., Kakoty, N. M., and Deka, D. (2017). A review of underwater robotics, navigation, sensing techniques and applications. In *Proceedings of the Advances in Robotics*, AIR '17, pages 8:1–8:6, New York, NY, USA. ACM.

[Clemens, 2017] Clemens, J. (2017). Multi-robot in-ice localization using graph optimization. In *IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, pages 36–42. IEEE.

[Cully et al., 2015] Cully, A., Clune, J., Tarapore, D., and Mouret, J. (2015). Robots that can adapt like animals. *Nature*, 521:503–507.

[Della-Maggiore et al., 2015] Della-Maggiore, V., Landi, S. M., and Villalta, J. I. (2015). Sensorimotor adaptation: Multiple forms of plasticity in motor circuits. *The Neuroscientist*, 21(2):109–125. PMID: 25122611.

[Doeller et al., 2010] Doeller, C., Barry, C., and Burgess, N. (2010). Evidence for grid cells in a human memory network. *Nature*, 463:657–61.

[Dorri et al., 2018] Dorri, A., Kanhere, S. S., and Jurdak, R. (2018). Multi-agent systems: A survey. *IEEE Access*, 6:28573–28593.

[Eliazar and Parr, 2004] Eliazar, A. I. and Parr, R. (2004). Learning probabilistic motion models for mobile robots. In *Proceedings of the Twenty-first International Conference on Machine Learning*, ICML '04, page 32, New York, NY, USA. ACM.

[Ernst and Blthoff, 2004] Ernst, M. and Blthoff, H. (2004). Merging the senses into a robust percept. *Trends in cognitive sciences*, 8(4):162–9.

[Estlin et al., 2007] Estlin, T., Gaines, D., Chouinard, C., Castano, R., Bornstein, B., Judd, M., Nesnas, I., and Anderson, R. (2007). Increased mars rover autonomy using ai planning, scheduling and execution. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 4911–4918.

[Fadiga et al., 2000] Fadiga, L., Fogassi, L., Gallese, V., and Rizzolatti, G. (2000). Visuomotor neurons: Ambiguity of the discharge or 'motor' perception? *International journal of psychophysiology : official journal of the International Organization of Psychophysiology*, 35(2–3):165–77.

[Faisal et al., 2008] Faisal, A., Selen, L., and Wolpert, D. (2008). Noise in the nervous system. *Nature reviews. Neuroscience*, 9(4):292–303.

[Feron and Johnson, 2008] Feron, E. and Johnson, E. N. (2008). *Springer Handbook of Robotics*, chapter Aerial Robotics, pages 1009–1029. Springer Berlin Heidelberg.

[Franz and Mallot, 2000] Franz, M. O. and Mallot, H. A. (2000). Biomimetic robot navigation. *Robotics and Autonomous Systems*, 30(1):133 – 153.

[Franz et al., 1998] Franz, M. O., Schölkopf, B., Mallot, H. A., and Bülthoff, H. H. (1998). Learning view graphs for robot navigation. *Autonomous Robots*, 5(1):111–125.

[Fraundorfer et al., 2007] Fraundorfer, F., Engels, C., and Nistér, D. (2007). Topological mapping, localization and navigation using image collections. *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3872–3877.

[Frese et al., 2010] Frese, U., Wagner, R., and Röfer, T. (2010). A slam overview from a user's perspective. *KI - Künstliche Intelligenz*, 24(3):191–198.

[Fyhn et al., 2004] Fyhn, M., Molden, S., Witter, M. P., Moser, E. I., and Moser, M.-B. (2004). Spatial representation in the entorhinal cortex. *Science*, 305(5688):1258–64.

[Ghysen, 2003] Ghysen, A. (2003). The origin and evolution of the nervous system. *The International journal of developmental biology*, 47(7-8):555–62.

[Graziano, 1999] Graziano, M. (1999). Where is my arm? the relative role of vision and proprioception in the neuronal representation of limb position. *Proceedings of the National Academy of Sciences of the United States of America*, 96:10418–21.

[Grimme et al., 2017] Grimme, A., Clemens, J., and Wille, R. (2017). Formal methods for reasoning and uncertainty reduction in evidential grid maps. *International Journal of Approximate Reasoning*, 87:23–39.

[Grisetti et al., 2010] Grisetti, G., Kmmerle, R., Stachniss, C., and Burgard, W. (2010). A tutorial on graph-based slam. *IEEE Transactions on Intelligent Transportation Systems Magazine*, 2:31–43.

[Guanella et al., 2007] Guanella, A., Kiper, D., and Verschure, P. (2007). A model of grid cells based on a twisted torus topology. *International journal of neural systems*, 17:231–40.

[H. Jacky Chang et al., 2007] H. Jacky Chang, C. S. George Lee, Y. Charlie Hu, and Yung-Hsiang Lu (2007). Multi-robot slam with topological/metric maps. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1467–1472.

[Hafting et al., 2005] Hafting, T., Fyhn, M., Molden, S., Moser, M.-B., and Moser, E. (2005). Microstructure of a spatial map in the entorhinal cortex. *Nature*, 436:801–6.

[Havoutis and Ramamoorthy, 2010a] Havoutis, I. and Ramamoorthy, S. (2010a). Constrained geodesic trajectory generation on learnt skill manifolds. *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2670–2675.

[Havoutis and Ramamoorthy, 2010b] Havoutis, I. and Ramamoorthy, S. (2010b). Geodesic trajectory generation on learnt skill manifolds. In *2010 IEEE International Conference on Robotics and Automation*, pages 2946–2952.

[Helwig et al., 2013] Helwig, S., Branke, J., and Mostaghim, S. (2013). Experimental analysis of bound handling techniques in particle swarm optimization. *IEEE Transactions on Evolutionary Computation*, 17(2):259–271.

[Hertzberg et al., 2011] Hertzberg, C., Wagner, R., Frese, U., and Schröder, L. (2011). Integrating generic sensor fusion algorithms with sound state representations through encapsulation of manifolds. *CoRR*, abs/1107.1119.

[Huang et al., 2017] Huang, Z., Wan, C., Probst, T., and Van Gool, L. (2017). Deep learning on lie groups for skeleton-based action recognition. In *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1243–1252.

[Huo and Smith, 2008] Huo, X. and Smith, A. (2008). A survey of manifold-based learning methods. *Recent Advances in Data Mining of Enterprise Data*, pages 691–745.

[Huston and Krapp, 2008] Huston, S. J. and Krapp, H. G. (2008). Visuomotor transformation in the fly gaze stabilization system. *PLoS Biology*, 6(7):e173.

[Imran et al., 2013] Imran, M., Hashim, R., and Khalid, N. E. A. (2013). An overview of particle swarm optimization variants. *Procedia Engineering*, 53:491 – 496. Malaysian Technical Universities Conference on Engineering & Technology 2012, MUCET 2012.

[Ingrand et al., 2007] Ingrand, F., Lacroix, S., Lemai-Chenevier, S., and Py, F. (2007). Decisional autonomy of planetary rovers: Research articles. *Journal of Field Robotics*, 24:559–580.

[Jaillet and Porta, 2013] Jaillet, L. and Porta, J. M. (2013). Path planning under kinematic constraints by rapidly exploring manifolds. *IEEE Transactions on Robotics*, 29(1):105–117.

[Jarvis et al., 2013] Jarvis, S., Worley, D., Hogy, S., Hill, A., Haussler, K., and Reiser, R. (2013). Kinematic and kinetic analysis of dogs during trotting after amputation of a thoracic limb. *American Journal of Veterinary Research*, 74(9):1155–1163.

[Jeffery and Burgess, 2006] Jeffery, K. J. and Burgess, N. (2006). A metric for the cognitive map: found at last? *Trends in Cognitive Sciences*, 10(1):1–3.

[Jékely, 2011] Jékely, G. (2011). Origin and early evolution of neural circuits for the control of ciliary locomotion. *Proceedings. Biological sciences*, 278(1707):914–922.

[Jonsson et al., 2000] Jonsson, A., Morris, P., Muscettola, N., Rajan, K., and Smith, B. (2000). Planning in interplanetary space: Theory and practice. In *Proceedings of the Fifth International Conference on Artificial Intelligence Planning Systems*, pages 177–186.

[Kelley, 1999] Kelley, C. T. (1999). *Iterative Methods for Optimization*. Society for Industrial and Applied Mathematics.

[Kennedy and Eberhart, 1995] Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of ICNN'95 - International Conference on Neural Networks*, volume 4, pages 1942–1948.

[Kerzel et al., 2001] Kerzel, D., Hommel, B., and Bekkering, H. (2001). A simon effect induced by induced motion and location: Evidence for a direct linkage of cognitive and motor maps. *Perception & Psychophysics*, 63(5):862–874.

[Kluss et al., 2015] Kluss, T., Marsh, W., Zetzsche, C., and Schill, K. (2015). Representation of impossible worlds in the cognitive map. *Cognitive processing*, 16(1):271–276.

[Kobilarov et al., 2009] Kobilarov, M., Crane, K., and Desbrun, M. (2009). Lie group integrators for animation and control of vehicles. *ACM Transactions on Graphics*, 28(2).

[Kohonen, 1982] Kohonen, T. (1982). Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43(1):59–69.

[Kuipers, 1982] Kuipers, B. (1982). The "map in the head" metaphor. In *Environment and Behavior*, volume 14, pages 202–220.

[Kuipers, 1983] Kuipers, B. (1983). *Spatial Orientation: Theory, Research, and Application*, chapter The Cognitive Map: Could It Have Been Any Other Way?, pages 345–359. Springer US.

[Kuipers, 2000] Kuipers, B. (2000). The spatial semantic hierarchy. *Artificial Intelligence*, 119(1):191–233.

[Kuipers and Beeson, 2002] Kuipers, B. and Beeson, P. (2002). Bootstrap learning for place recognition. In *Eighteenth National Conference on Artificial Intelligence*, pages 174–180, Menlo Park, CA, USA. American Association for Artificial Intelligence.

[Kuipers et al., 2006] Kuipers, B., Beeson, P., Modayil, J., and Provost, J. (2006). Bootstrap learning of foundational representations. *Connect. Sci.*, 18:145–158.

[Kuipers and Levitt, 1988] Kuipers, B. and Levitt, T. (1988). Navigation and mapping in large scale space. *AI Magazine*, 9:25–43.

[Laflaquire et al., 2015] Laflaquire, A., ORegan, J. K., Argentieri, S., Gas, B., and Terekhov, A. V. (2015). Learning agents spatial configuration from sensorimotor invariants. *Robotics and Autonomous Systems*, 71:49–59. Emerging Spatial Competences: From Machine Perception to Sensorimotor Intelligence.

[Lang et al., 2007] Lang, T., Plagemann, C., and Burgard, W. (2007). Adaptive non-stationary kernel regression for terrain modeling. In *Robotics: Science and Systems III*. Robotics: Science and Systems Foundation.

[Lee, 2003] Lee, J. (2003). *Introduction to Smooth Manifolds*. Graduate Texts in Mathematics. Springer New York.

[Leonard and Durrant-Whyte, 1991] Leonard, J. J. and Durrant-Whyte, H. F. (1991). Mobile robot localization by tracking geometric beacons. *IEEE Transactions on Robotics and Automation*, 7(3):376–382.

[Lin and Zha, 2008] Lin, T. and Zha, H. (2008). Riemannian manifold learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(5):796–809.

[Mallot et al., 1995] Mallot, H., Blthoff, H., Georg, P., Schlkopf, B., and Yasuhara, K. (1995). View-based cognitive map learning by an autonomous robot. In *Proceedings of the International Conference on Artificial Neural Networks*, pages 381–386.

[Marjaninejad et al., 2019] Marjaninejad, A., Urbina-Melndez, D., Cohn, B., and Valero-Cuevas, F. (2019). Autonomous functional movements in a tendon-driven limb via limited experience. *Nature Machine Intelligence*, 1:144–154.

[McCulloch and Pitts, 1943] McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133.

[Mcnaughton et al., 2006] Mcnaughton, B., Battaglia, F., Jensen, O., Moser, E., and Moser, M.-B. (2006). Path integration and the neural basis of the cognitive map.. *Nature reviews. Neuroscience*, 7:663–78.

[Mhatre and Gorchetchnikov, 2012] Mhatre, H. and Gorchetchnikov, A. (2012). Grid cell hexagonal patterns formed by fast self-organized learning within entorhinal cortex. *Hippocampus*, 22:320–34.

[Milford et al., 2004] Milford, M. J., Wyeth, G. F., and Prasser, D. (2004). Ratslam: a hippocampal model for simultaneous localization and mapping. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, volume 1, pages 403–408 Vol.1.

[Mirzaei et al., 2015] Mirzaei, H., Zarbafian, S., Villar, E. A., Mottarella, S. E., Beglov, D., Vajda, S., Paschalidis, I. C., Vakili, P., and Kozakov, D. (2015). Energy minimization on manifolds for docking flexible molecules. *Journal of chemical theory and computation*, 11(3):1063–76.

[Muscettola et al., 1998] Muscettola, N., Nayak, P. P., Pell, B., and Williams, B. C. (1998). Remote agent: To boldly go where no ai system has gone before. *Artificial. Intelligence*, 103(1-2):5–47.

[Najjar and Hasegawa, 2013] Najjar, T. and Hasegawa, O. (2013). Self-organizing incremental neural network (soinn) as a mechanism for motor babbling and sensory-motor learning in developmental robotics. In Rojas, I., Joya, G., and Gabestany, J., editors, *Advances in Computational Intelligence*, pages 321–330, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Nakath et al., 2017] Nakath, D., Clemens, J., and Rachuy, C. (2017). Rigid body attitude control based on a manifold representation of direction cosine matrices. *Journal of Physics: Conference Series*, 783(1):012040.

[Nakath et al., 2019] Nakath, D., Clemens, J., and Rachuy, C. (2019). Active asteroid-slam - active graph slam with landing site discovery in a deep space proximity operations scenario. *Journal of Intelligent & Robotic Systems*.

[Nakath et al., 2014] Nakath, D., Kluth, T., Reineking, T., Zetzsche, C., and Schill, K. (2014). Active sensorimotor object recognition in three-dimensional space. In Freksa, C., Nebel, B., Hegarty, M., and Barkowsky, T., editors, *Spatial Cognition IX*, pages 312–324. Springer International Publishing.

[Nakath et al., 2016] Nakath, D., Rachuy, C., Clemens, J., and Schill, K. (2016). Optimal rotation sequences for active perception. In Braun, J. J., editor, *Proc. SPIE: Multisensor, Multisource Information Fusion: Architectures, Algorithms, and Applications*, volume 9872, pages 987204-1–987204-13. SPIE Press.

[Nocedal and Wright, 2006] Nocedal, J. and Wright, S. (2006). *Numerical optimization*. Springer series in operations research and financial engineering. Springer New York, 2nd edition.

[Nordkvist and Sanyal, 2010] Nordkvist, N. and Sanyal, A. K. (2010). A lie group variational integrator for rigid body motion in se(3) with applications to underwater vehicle dynamics. In *49th IEEE Conference on Decision and Control (CDC)*, pages 5414–5419.

[O'Keefe et al., 1998] O'Keefe, J., Burgess, N., Donnett, J., Jeffery, K., and Maguire, E. (1998). Place cells, navigational accuracy, and the human hippocampus. *Philosophical transactions of the Royal Society of London. Series B, Biological sciences*, 353(1373):1333–40.

[O'Keefe and Nadel, 1978] O'Keefe, J. and Nadel, L. (1978). *The hippocampus as a cognitive map*. Clarendon Press, Oxford, United Kingdom.

[Olsson et al., 2005] Olsson, L., Nehaniv, C., and Polani, D. (2005). From unknown sensors and actuators to visually guided movement. In *Proceedings. The 4th International Conference on Development and Learning*.

[Olsson et al., 2006] Olsson, L., Nehaniv, C. L., and Polani, D. (2006). From unknown sensors and actuators to actions grounded in sensorimotor perceptions. *Connect. Sci.*, 18:121–144.

[ORegan and No, 2001] ORegan, J. and No, A. (2001). A sensorimotor account of vision and visual consciousness. *Behavioral and Brain Sciences*, 24:939–973.

[Parsopoulos, 2016] Parsopoulos, K. E. (2016). *Handbook of Heuristics*, chapter Particle Swarm Methods, pages 1–47. Springer International Publishing.

[Pedraza et al., 2009] Pedraza, L., Rodrguez-Losada, D., Matia, F., Dissanayake, G., and Valls Miro, J. (2009). Extending the limits of feature-based slam with b-splines. *Robotics, IEEE Transactions on*, 25:353–366.

[Perlin, 2002] Perlin, K. (2002). Improving noise. *ACM Trans. Graph.*, 21(3):681–682.

[Philipona et al., 2003] Philipona, D., O'Regan, J. K., and Nadal, J. . (2003). Is there something out there? inferring space from sensorimotor dependencies. *Neural Computation*, 15(9):2029–2049.

[Philipona et al., 2004] Philipona, D., regan, J. O., p. Nadal, J., and Coenen, O. (2004). Perception of the structure of the physical world using unknown multimodal sensors and effectors. In Thrun, S., Saul, L. K., and Schölkopf, B., editors, *Advances in Neural Information Processing Systems 16*, pages 945–952. MIT Press.

[Piaget, 1953] Piaget, J. (1953). *The origin of Intelligence in the Child*. Routledge & Kegan, London.

[Pierce and Kuipers, 1997] Pierce, D. and Kuipers, B. (1997). Map learning with uninterpreted sensors and effectors. *Artificial Intelligence*, 92:169–227.

[Pitelis et al., 2013] Pitelis, N., Russell, C., and Agapito, L. (2013). Learning a manifold as an atlas. In *Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1642–1649.

[Plice et al., 2003] Plice, L., Lau, B., Pisanich, G., and Young, L. A. (2003). Biologically inspired behavioral strategies for autonomous aerial explorers on mars. In *2003 IEEE Aerospace Conference Proceedings (Cat. No.03TH8652)*, volume 1, pages 1–304 vol.1.

[Poli et al., 2007] Poli, R., Kennedy, J., and Blackwell, T. (2007). Particle swarm optimization: An overview. *Swarm Intelligence*, 1:33–57.

[Prassler and Kosuge, 2008] Prassler, E. and Kosuge, K. (2008). *Springer Handbook of Robotics*, chapter Domestic Robotics, pages 1253–1281. Springer Berlin Heidelberg.

[Probst et al., 2015] Probst, A., Peytav, G. G., Nakath, D., Schattel, A., Rachuy, C., Lange, P., Clemens, J., Echim, M., Schwarting, V., Srinivas, A., Gadzicki, K., Frstner, R., Eissfeller, B., Schill, K., Bskens, C., and Zachmann, G. (2015). Kanaria: Identifying the challenges for cognitive autonomous navigation and guidance for missions to small planetary bodies. In *66th International Astronautical Congress (IAC)*, Jerusalem, Israel.

[Provost et al., 2006] Provost, J., Kuipers, B., and Miikkulainen, R. (2006). Developing navigation behavior through self-organizing distinctive-state abstraction. *Connection Science*, 18(2):159–172.

[Ramos and Ott, 2016] Ramos, F. and Ott, L. (2016). Hilbert maps: Scalable continuous occupancy mapping with stochastic gradient descent. *The International Journal of Robotics Research*, 35:1717–1730.

[Rasmussen and Williams, 2005] Rasmussen, C. E. and Williams, C. K. I. (2005). *Gaussian Processes for Machine Learning*. The MIT Press, London, England.

[Reineking et al., 2010] Reineking, T., Wolter, J., Gadzicki, K., and Zetzsche, C. (2010). Bio-inspired architecture for active sensorimotor localization. In *Proceedings of the 7th International Conference on Spatial Cognition*, SC'10, pages 163–178. Springer Berlin, Heidelberg.

[Remolina and Kuipers, 1998] Remolina, E. and Kuipers, B. (1998). Towards a formalization of the spatial semantic hierarchy. *Fourth Symposium on Logical Formalizations of Commonsense Reasoning*.

[Rojas, 1996] Rojas, R. (1996). *Neural Networks: A Systematic Introduction*. Springer Berlin, Heidelberg.

[Russell and Norvig, 2003] Russell, S. J. and Norvig, P. (2003). *Artificial Intelligence: A Modern Approach*. Pearson Education, New Jersey, 2nd edition.

[SAE International, 2018] SAE International (2018). *Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles*. SAE International.

[Saegusa et al., 2009] Saegusa, R., Metta, G., Sandini, G., and Sakka, S. (2009). Active motor babbling for sensory-motor learning. In *2008 IEEE International Conference on Robotics and Biomimetics*, pages 794 – 799.

[Sarti et al., 1993] Sarti, A., Walsh, G., and Sastry, S. (1993). Steering left-invariant control systems on matrix lie groups. In *Proceedings of 32nd IEEE Conference on Decision and Control*, pages 3117–3121 vol.4.

[Schill, 1997] Schill, K. (1997). Decision support systems with adaptive reasoning strategies. In Freksa, C., Jantzen, M., and Valk, R., editors, *Foundations of Computer Science*, volume 1337, pages 417–427. Springer Science + Business Media, Berlin/Heidelberg.

[Schiller et al., 2015] Schiller, D., Eichenbaum, H., Buffalo, E. A., Davachi, L., Foster, D. J., Leutgeb, S., and Ranganath, C. (2015). Memory and space: Towards an understanding of the cognitive map. *Journal of Neuroscience*, 35(41):13904–13911.

[Selig, 2007] Selig, J. (2007). *Geometric Fundamentals of Robotics*. Monographs in Computer Science. Springer New York.

[Senanayake and Ramos, 2017] Senanayake, R. and Ramos, F. (2017). Bayesian hilbert maps for dynamic continuous occupancy mapping. In *Proceedings of the 1st Annual Conference on Robot Learning*, volume 78, pages 458–471.

[Shafer, 1976] Shafer, G. (1976). *A Mathematical Theory of Evidence*. Princeton University Press, Princeton.

[Siegel and White, 1975] Siegel, A. W. and White, S. H. (1975). The development of spatial representations of large-scale environments. In *Advances in Child Development and Behavior*, volume 10, pages 9–55. JAI.

[Siegwart and Nourbakhsh, 2004] Siegwart, R. and Nourbakhsh, I. (2004). *Introduction to Autonomous Mobile Robots*. Bradford Book.

[Silver, 2010] Silver, R. (2010). Neuronal arithmetic. *Nature reviews. Neuroscience*, 11:474–89.

[Solstad et al., 2009] Solstad, T., Boccara, C., Kropff, E., Moser, M.-B., and Moser, E. (2009). Representation of geometric borders in the entorhinal cortex. *Science (New York, N.Y.)*, 322:1865–8.

[Starek et al., 2016] Starek, J., AkmeÅe, B., Nesnas, I., and Pavone, M. (2016). Spacecraft autonomy challenges for next-generation space missions. *Lecture Notes in Control and Information Sciences*, 460.

[Stillwell, 2008] Stillwell, J. (2008). *Naive Lie Theory*. Undergraduate Texts in Mathematics. Springer New York.

[Sturm et al., 2008] Sturm, J., Plagemann, C., and Burgard, W. (2008). Unsupervised body scheme learning through self-perception. In *2008 IEEE International Conference on Robotics and Automation*, pages 3328–3333.

[Taube et al., 1990a] Taube, J., Muller, R., and Ranck, J. (1990a). Head-direction cells recorded from the postsubiculum in freely moving rats. I. description and quantitative analysis. *Journal of Neuroscience*, 10(2):420–435.

[Taube et al., 1990b] Taube, J., Muller, R., and Ranck, J. (1990b). Head-direction cells recorded from the postsubiculum in freely moving rats. II. effects of environmental manipulations. *Journal of Neuroscience*, 10(2):436–447.

[Teodoro et al., 2001] Teodoro, M. L., Phillips, G. N., and Kavraki, L. E. (2001). Molecular docking: a problem with thousands of degrees of freedom. *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No.01CH37164)*, 1:960–965 vol.1.

[Thrun, 2002] Thrun, S. (2002). Robotic mapping: A survey. In Lakemeyer, G. and Nebel, B., editors, *Exploring Artificial Intelligence in the New Millenium*. Morgan Kaufmann.

[Thrun, 2003] Thrun, S. (2003). Learning occupancy grid maps with forward sensor models. *Autonomous Robots*, 15(2):111–127.

[Thrun and Bücken, 1996] Thrun, S. and Bücken, A. (1996). Integrating grid-based and topological maps for mobile robot navigation. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence - Volume 2*, AAAI'96, pages 944–950. AAAI Press.

[Thrun et al., 2005] Thrun, S., Burgard, W., and Fox, D. (2005). *Probabilistic Robotics*. MIT press, Cambridge, MA.

[Tolman, 1948] Tolman, E. C. (1948). Cognitive maps in rats and men. *Psychological Review*, 55(4):189–208.

[Ulanovsky, 2011] Ulanovsky, N. (2011). Neuroscience: How is three-dimensional space encoded in the brain? *Current Biology*, 21(21):R886 – R888.

[Urmson and Whittaker, 2008] Urmson, C. and Whittaker, W. (2008). Self-driving cars and the urban challenge. *IEEE Intelligent Systems*, 23(2):66–68.

[Visatemongkolchai and Zhang, 2007] Visatemongkolchai, A. and Zhang, H. (2007). Building probabilistic motion models for slam. In *2007 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 1629–1634.

[Walsh et al., 1994] Walsh, G., Montgomery, R., and Sastry, S. (1994). Optimal path planning on matrix lie groups. In *Proceedings of the 33rd Conference on Decision and Control*, page 12581263.

[Warren et al., 2017] Warren, W. H., Rothman, D. B., Schnapp, B. H., and Ericson, J. D. (2017). Wormholes in virtual space: From cognitive maps to cognitive graphs. *Cognition*, 166:152 – 163.

[Welch et al., 2013] Welch, R., Limonadi, D., and Manning, R. (2013). Systems engineering the curiosity rover: A retrospective. In *Proceedings of the 8th International Conference on System of Systems Engineering*, pages 70–75.

[Wexler et al., 1998] Wexler, M., Kosslyn, S. M., and Berthoz, A. (1998). Motor processes in mental rotation. *Cognition*, 68(1):77 – 94.

[Whitney, 1944] Whitney, H. (1944). The self-intersections of a smooth n-manifold in 2n-space. *Annals of Mathematics*, 45(2):220–246.

[Williams et al., 2005] Williams, A., Barrus, S., Morley, R. K., and Shirley, P. (2005). An efficient and robust ray-box intersection algorithm. In *ACM SIGGRAPH 2005 Courses*, SIGGRAPH '05, New York, NY, USA. ACM.

[Wooldridge, 2000] Wooldridge, M. (2000). *Reasoning About Rational Agents*. MIT Press.

[Wu et al., 2012] Wu, Y., Wang, H., Zhang, B., and Du, K.-L. (2012). Using radial basis function networks for function approximation and classification. In *ISRN Applied Mathematics*.

[Yartsev and Ulanovsky, 2013] Yartsev, M. M. and Ulanovsky, N. (2013). Representation of three-dimensional space in the hippocampus of flying bats. *Science*, 340(6130):367–372.

[Zetzsche et al., 2008] Zetzsche, C., Wolter, J., and Schill, K. (2008). Sensorimotor representation and knowledge-based reasoning for spatial exploration and localisation. *Cognitive Processing*, 9(4):283–297.